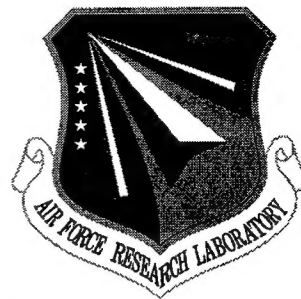AFRL-SN-RS-TR-1999-48, Vol I (of two)
Final Technical Report
May 1999

MCARM/STAP DATA ANALYSIS

Research Associates for Defense Conversion, Inc.

Vincent Cavo

19990629 085

**AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

DTIC QUALITY INSPECTED 4

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1999-48, Vol I (of two) has been reviewed and is approved for publication.

APPROVED: *[signature]* TODD B. HALE

TODD B. HALE, Capt, USAF
Project Engineer

FOR THE DIRECTOR: *[signature]* Robert G. Polce

ROBERT G. POLCE, Acting Chief
Rome Operations Office
Sensors Directorate

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE May 1999	3. REPORT TYPE AND DATES COVERED Final Apr 96 - Feb 98	

4. TITLE AND SUBTITLE MCARM/STAP DATA ANALYSIS, Volume I (of two)	5. FUNDING NUMBERS C - F30602-96-C-0053 PE - 62702F PR - 4506
6. AUTHOR(S) Vincent Cavo	TA - 11 WU - 2K

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research Associates for Defense Conversion, Inc. 10002 Hillside Terrace Marcy NY 13403	8. PERFORMING ORGANIZATION REPORT NUMBER N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/SNRT 26 Electronics Parkway Rome NY 13441-4514	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-SN-RS-TR-1999-48, Volume I (of two)

11. SUPPLEMENTARY NOTES

Air Force Research Laboratory Project Engineer: Captain Todd Hale/SNRT/(315) 330-1896

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Multi-Channel Airborne Radar Measurement (MCARM) program was initiated to gather real airborne data. Both monostatic and bistatic data were collected, under varying environmental conditions and in different geographical areas. This report documents the construction, maintenance, and use of the MCARM data via the Internet. The main objective of this access is to permit and promote research in the area of Space-Time Adaptive Processing. Use of real airborne data will permit researchers to better test different signal processing algorithms and hypotheses.

A query form and query engine permits a user to search the data repository. The data repository is comprised of 570 data files and occupies approximately 27 gigabytes of magnetic disk storage.

14. SUBJECT TERMS MCARM, STAP, Adaptive Processing, Signal Processing, Data Processing			15. NUMBER OF PAGES 86
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Table of Contents

Section Title   **Page**

List of Figures

List of Tables

Appendices

Glossary of Terms

Term	Definition
CATIS	Computer Aided Tactical Information System
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off –The-Shelf
CPI	Coherent Processing Interval
CREST	Common Research Environment for STAP Technology
DoD	Department of Defense
ESRI	Environmental Systems Research Institute
GMTI	Ground Moving Target Indicator
GPS	Global Positioning System
HiPPI	High Performance Parallel Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HPCC	High Performance Computer Center
IE 2000	Image Exploitation 2000 server
I/Q	In-phase and Quadrature Phase
MCARM	Multi-Channel Airborne Radar Measurement
MHPCC	Maui High Performance Computer Center
NCSA	National Center for Supercomputing Applications
OMG	Object Management Group
OSF	Open Software Foundation
POWER	Performance Optimized With Enhanced RISC
PRF	Pulse Repetition Frequency
RDT&E	Research Development Test & Evaluation
RISC	Reduced Instruction Set Computing
RLSTAP/ADT	Rome Laboratory Space-Time Adaptive Processing/Algorithm Development Tool
RSTER	Radar Surveillance Technology Experimental Radar
SCSI	Small Computer Systems Interface
SP2	Scalable POWERparallel 2
STAP	Space-Time Adaptive Processing
TCP/IP	Transmission Control Protocol/Internet Protocol
UTM	Universal Transverse Mercator
VIFF	Khoros Visualization Image File Format
VRML	Virtual Reality Modeling Language
WWW	World Wide Web

1.0 Executive Summary

The Multi-Channel Airborne Radar Measurement (MCARM) program was initiated to gather real airborne data. The MCARM data collection process involved the acquisition of both monostatic and bistatic data under varying environmental conditions and in different geographical areas. Real airborne data permits researchers to better test different signal processing algorithms and hypotheses. Making the data available via the Internet permits researchers throughout the United States and in other countries to participate in experimentation.

This Final Report documents the data acquisition process, via the Internet, assesses the quality of the data, and discusses some possible analysis that can be performed by researchers. The format of the MCARM data is publicly provided. However, except for some sample files, access to the data is restricted and available to only authorized personnel.

The Final Report for this effort is divided into two parts. To facilitate comprehension, the report is divided into a data acquisition part (Part I) and data analysis part (Part II). Each part addresses the relevant Statement of Work (SOW) tasks.

2.0 Introduction

The Multi-Channel Airborne Radar Measurement (MCARM) program was initiated to gather real airborne data. The MCARM data collection process involved the acquisition of both monostatic and bistatic data under varying environmental conditions and in different geographical areas.

As a direct result of this effort, the MCARM data files have been made available to researchers via the Internet. The current MCARM data repository resides on a SparcServer 630 MP. The data repository consists of 570 preprocessed data files; all are available via the Internet through Hyper Text Transfer Protocol (HTTP). A query form and query engine permits a user to search the data repository on any valid MCARM parameter.

The data repository occupies approximately 27 gigabytes of magnetic disk storage. A data search on a particular parameter may take four minutes or more. The search time is dependent upon the system and network load. Access to the MCARM data is restricted except for some publicly available files.

This part of the Final Report documents work performed in making the MCARM data available via the Internet. This effort makes data available by replicating and building upon the Common Research Environment for STAP Technology (CREST) environment, the Image Exploitation 2000 (IE 2000) server and data base engine, and the MCARM data.

The Final Report for this effort is divided into two parts. To facilitate comprehension, the report is divided into a data acquisition part (Part I) and data analysis part (Part II). Each part addresses the relevant Statement of Work (SOW) tasks.

3.0 Background

This section discusses background information pertinent to this effort. More specifically, the section addresses the CREST environment, the IE 2000 system, and the MCARM data.

3.1 CREST

In essence, the CREST environment is a radar data repository and the accompanying software and hardware tools needed to process data within the repository. The environment provides the means to enable STAP researchers to develop and process algorithms using the Rome Laboratory Space-Time Adaptive Processing (RLSTAP) tool. Meanwhile, the CREST Data Library provides data that can be used to test the algorithms. The CREST environment is dependent on the Internet for communications and upon the resources of MHPCC for data processing.

The RLSTAP/Algorithm Development Tool (ADT) is a user-friendly software environment for STAP simulations, processing and analysis. This tool represents a complete radar environment consisting of simulated and measured data, a full library of signal processing routines, and a flexible graphical interface. The RLSTAP/ADT is a Unix X-Windows client application currently running on Sun (SunOS 4.1.3, Solaris2.4) and RS6000 (AIX V3.2) platforms. The tool permits a user to prototype and perform experiments graphically. A user can either test algorithms with their own data or use the CREST Data Library to acquire RSTER data. Experiments requiring extensive computing power can be batched remotely to the MHPCC. (The computing power at the MHPCC is supplied by an IBM POWER parallel SP2 System comprised of 400 Power2 nodes linked by a high performance switch. The MHPCC offers a 100 gigaflop capability for 100 billion floating point operations per second.)

The CREST environment executes over the Internet via TCP/IP. The Data Library currently utilizes the WWW http protocol while the Remote-RLSTAP/ADT uses TCP/IP based applications. The client environments utilize low bandwidth message transfers to the servers at the MHPCC.

3.2 IE 2000

A number of server concepts were borrowed from the Imagery Exploitation (IE) 2000 during the construction of the CREST Data Library. The IE 2000 is a software facility developed by Rome Laboratory to provide an experimental digital image exploitation test bed for development and evaluation of functions, techniques, software, and hardware to support imagery exploitation. "The facility is composed of image processing hardware and software that provide the digital image exploitation environment required for supporting the development of automated target detection, identification, location, as well as the experimental production of advanced target materials. The facility was developed using an open system architecture."

3

The facility allows digital image interpretation on multisensor/multispectral digital imagery using low cost soft copy workstations. The system also uses geographic information, cartographic data, and an intelligence data base as aids to the analyst. The available hardware and software provide a full range of digital image processing tools such as image enhancements, warping, image registration, mensuration, etc. This facility represents the future in soft copy Reconnaissance Technical Unit's image interpretation functionality and capability.

The IE 2000 facility is equipped with twelve Sun Workstations, a Sun SPARC 1000 Server, a 39 gigabyte Alphatronix Optical Disk Jukebox, a VAX 11/785 Computer running Combat CATIS, Kodak PhotoCD production capability, high resolution image scanners, color printers and a wide range of operational public and state-of-the-art COTS image exploitation applications software. All image processing hardware is on a local area network with future planned fiber optic connectivity to other facilities throughout Rome Laboratory.

In brief, the IE 2000 is a good example and template of the environment for which the CREST Data Library would operate. The IE 2000 provided an excellent 'lessons-learned' baseline. Also, the IE 2000 was developed at Rome Laboratory and it is owned by the Air Force. These reasons provided justification to model the CREST Data Library after the IE 2000.

4.0 Tasks/Technical Requirements

This section describes the tasks outlined in the SOW, which pertain to the data acquisition process. Specifically, it addresses the work that had to be performed to make the MCARM data available via the Internet.

4.1 CREST Structure and Architectural Analysis

The CREST/RLSTAP ADT tool has taken many years to develop. At the time of this writing, a new contract has been awarded to incorporate new and advanced technology within this environment. As such, the environment is very dynamic, fluid and complex. The interfacing of new user data, as the MCARM data, is not often easy or straightforward. For example, the MCARM data must first be converted to the internal VIFF format and the header parameters properly mapped before the data can be utilized within the RLSTAP/ADT tool. Additionally, some researchers may resist the use of the RLSTAP/ADT processing paradigm. These researchers may instead choose procedures and mechanisms that they may be more accustomed and familiar with, such as the use of MCARM data in conjunction with Matlab.

A careful and thorough analysis of the CREST structure and architecture was performed at the outset of this effort. As a result of the analysis, it was concluded that the integration of the MCARM data within the CREST environment did not make sense at that time. The integration would have several adverse affects: 1) it would hinder ongoing development of CREST; 2) cooperation between the CREST/MCARM teams would be problematic due to the need for gradual integration (the MCARM data required further processing and receipt of the data from Westinghouse was incremental); 3) RL would lose control of the data; and 4) it would extend the amount of time required to create a MCARM data repository accessible via the Internet because of interaction between the CREST/MCARM teams. Consequently, it was more prudent to create a separate data repository that is accessible from CREST via a hypertext link or vice versa.

4.1.1 CREST Environment Replication

The current MCARM data repository replicates the CREST environment and architecture as much as is deemed appropriate. Any items that complicated the user interface, made visualization by the user more difficult, or were being altered at the time of implementation were eliminated. The main objective in constructing the user interface for the MCARM data was user-friendliness. For example, a user may download a given file by simply placing the cursor (via a pointing device) over the desired file and selecting it (e.g. by depressing a mouse button). (The file will begin to download after the appropriate authorization, if any.) User can also geographically display any particular flight path used during the data acquisition process. (Presently only flight number five can be geographically displayed.)

The architecture used to implement the MCARM data repository is depicted below:

NAVIGATION HIERARCHY
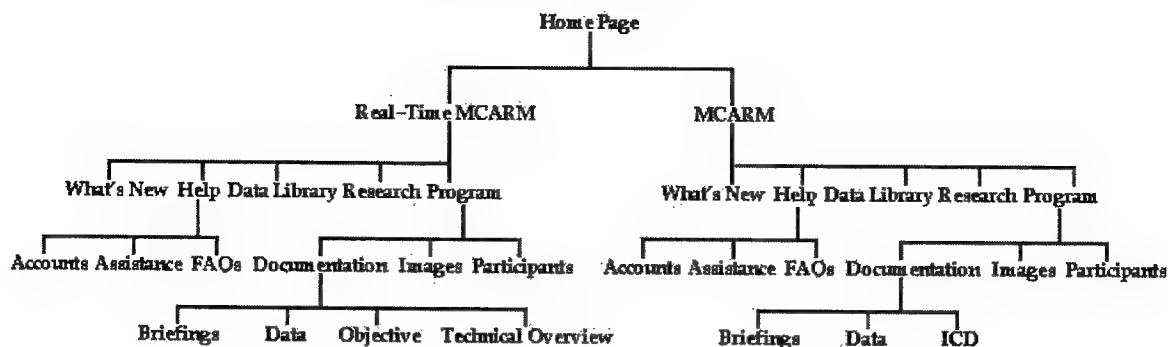
Figure 4.1.1-1 MCARM Web Hierarchy

Users navigate the MCARM hierarchy (i.e. the Web pages) by simply selecting any 'leaf' or page. Each 'leaf' within the hierarchy represents a link to a particular page within the above structure. Users perform most functions through this point-and-click mechanism while accessing the MCARM data via the Internet.

The above graphically depicted hierarchy is displayed in the 'beginner' mode (see Home Page) or within the frames mode. In the 'beginner' mode, users must scroll the page until the above graphical hierarchy is displayed before the MCARM Web pages can be navigated. In the frame mode (i.e. Frames is selected from the Home Page), the browser window splits to display the hierarchy at the bottom and the selected page at the top. In this mode, users must first select the branch of interest (Real-Time MCARM or MCARM) before the appropriate hierarchy is displayed. Then the appropriate page can be selected by moving the mouse cursor over the desired text (e.g. Accounts, Images, Briefings). Note that if a 'leaf' is not selected multiple page selections will be displayed (e.g. selection of Help will display a page containing the selections Accounts, Assistance, and FAQs). This requires more steps to arrive at the desired page.

4.1.2 IE 2000 Replication

Client-server concepts were borrowed from the IE 2000 for the implementation of the CREST Data Library. CREST Data Library makes existing radar data widely accessible via the Internet to researchers. For the purposes of MCARM, the inherent client/server nature of the Internet was adequate for the dissemination of its data. A number of free software tools were identified and acquired through the Internet to manage and control access to the MCARM data. Due to time constraints, the sophisticated object-oriented design planned for the CREST Data Library (i.e. Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA)) was not implemented. It was not clear what benefit the object-oriented design would provide for the MCARM data access. However, it was clear that complexity would be a direct outcome. Thus, the Internet MCARM implementation used basic concepts already provided by the Web.

6

4.2 CREST Enhancement and Modification For Monostatic MCARM Data

Because of reasons stated previously in Section 4.1 it was determined that it would be more prudent to create a separate Web page for MCARM, which would be accessible from CREST. Hence, this section discusses work performed to create the MCARM Web site and permit access from/to CREST.

4.2.1 Develop Internet/WWW Procedures

Procedures were established to make the MCARM data available via the Internet/WWW. This entailed the creation of a framework for the data, the construction of Web pages, the acquisition of public domain software tools, the establishment of user access mechanisms, and the development of query software.

A framework was created to permit access to MCARM and real-time MCARM data and information. This framework is depicted in Section 4.1.1 above. Then, HTML-based Web pages were constructed to permit access to the MCARM information and data. Construction of the Web pages involved the creation of image maps and graphics using public domain or system software tools.

Public domain software tools were located and acquired for the construction, maintenance, and administration of the MCARM Web server. Software tools (e.g. compression and decompression, browsers, viewers, helper applications, data security, report generation, format conversion (e.g. Postscript to HTML), animation, Web administration, etc.) were acquired, for various platforms, for downloading purposes by MCARM Internet users. For example, the 'tools' directory (http://sunrise.deepthought.rl.af.mil/tools/) contains browsers, viewers, and utilities for Sun, MacIntosh, and PCs. The PC software tools are for Windows for Work Groups, Windows NT, Windows 95, and Windows 3.1 clients. For the most part, the software tools 'xpaint', 'mapedit', 'vi' and 'hotmetal' were used for the construction and maintenance of the Web pages. The latter two were used to create the HTML for the Web pages.

Upon completion of the Web pages, user account mechanisms were put in place for the creation and management of user accounts. The purpose of these accounts is to restrict and control access to sensitive MCARM data. A prospective user requests an account by completing the appropriate online forms. Upon receipt, cognizant RL staff review the forms and an account request is either accepted or denied. For new accounts, a user is assigned a password and access privileges are setup. Table 4.2.1-1 below depicts the user accounts that were created during this effort. In some instances, changes also involve the creation of directory work areas for general and specific users. Software was written to receive and process the Internet forms data (see Appendix C – Accounts Request Form).

7

Organization	Individual
AFIT	Various Students
Australian DoD	Dr. Daniel Madurasinghe
GEC-Marconi UK	Requested
Hanscom AFB	Dr. McKee
LeHigh Univ	Prof. Rick Blum
LeHigh Univ	Zhongxiu Gu
Mitre	Dr. Suresh Babu
Northrup	Mr. Robert Warta
RADC	Dr. Braham Himed
RL/OCSA	Dr. Peter Zulch
RL/OCSM	Jim Michels/ Dennis Davis (Florida Contractor)
RL/OCSS	Dr. William Melvin
SRC	Dr. Harvey Shuman
Swedish Defense	Requested
Syracuse Univ	Prof. Tapan Sarkar
Syracuse Univ	Sheeyan Park
Univ. New Mexico	Prof. LeBlanc

Table 4.2.1-1 MCARM Internet User Accounts

Lastly, query software was written in C and an associated HTML query form was developed to permit MCARM data to be queried via the Internet. Through the query form a user may query the MCARM database for particular parameters and their associated value (e.g. a particular acquisition number.) It is also possible to specify a range of values for parameters. The query Web page also permits the display of geography and the associated flight paths.

4.2.2 Make Data Accessible Via Internet

To make the MCARM data available via the Internet it was necessary to: 1) identify, acquire, and install server software on an RL computer; 2) the appropriate security mechanisms had to put in place; 3) software had to be written to extract header information from the data; and 4) a data repository had to be constructed. As with CREST data, the MCARM data files were split into header and CPI components. This splitting of the data facilitates local searching and the transmission of the data over the Internet. Entire data files, both the header and CPI portions, are transmitted upon request. The header information is used only to qualify a search of the MCARM data repository.

A National Center for Supercomputing Applications (NCSA) HTTP Web server was identified, acquired, compiled and configured for Internet access of the MCARM data. Initially, this took place on a Sun server computer ('isaiah'). Subsequently, it was moved to another Sun platform

('sunrise' a SPARCserver 630 MP computer) due to disk storage, operating system, and administration reasons.

Security mechanisms were established for the access of the MCARM data via the Internet. Two levels of security were implemented to access the data. First, a user had to obtain a user account and password. Secondly, a user had to provide the IP address of the computer from which the data would be accessed. The user account and password was validated when a request for controlled data was made. Meanwhile, the IP address of the target computer was checked before the data was transmitted. Both of these security checks have to be satisfied before any data is released.

Upon completion of the above tasks an Internet browser (Netscape) was acquired to test the security mechanisms and MCARM data access via the World Wide Web. The initial tests, which proved successful, were performed with a limited amount of MCARM data. Some modifications to the software were required as the data repository grew. For example, the file naming convention had to be modified to retain compatibility with PCs.

It was recognized in the beginning that a true data base management system (DBMS), in lieu of the CREST based IE2000 DBMS, would be required as the MCARM data repository grew in size. Hence, a DBMS framework was investigated for the data repository to improve functionality, ease of use, and maintainability. A DBMS would automatically handle the administration, access and location of data files on the server (i.e. the location of data on different disk drives would be transparent).

4.2.3 Develop RLSTAP/MCARM Needs Assessment

The ability to access MCARM data from within the RLSTAP tool was a definite requirement. However, the parameter names used within RLSTAP were different than the parameter names used within MCARM. Both data files were in Matlab format. Hence, what was needed was a translation capability to convert the parameter names used in the MCARM data to names that were compatible with RLSTAP. A procedure was required to perform the mapping, read the MCARM header information and generate the VIFF file required by Khoros (Khoros is the underlying visual tool used by RLSTAP). This procedure permits the use of MCARM data within RLSTAP. The purpose of this section is to further discuss this procedure.

4.2.3.1 Design, Develop and Implement Interface

All that is required to access MCARM data from within RLSTAP is to have the same parameter names within the header. Hence, once created, a parameter translation table can be used to map MCARM parameters to parameter names used by RLSTAP. This translation table was created to perform the parameter mapping. The translation required the addition of new parameters to the MCARM header and the mapping of existing parameters to those required by RLSTAP. These steps were required because no one to one mapping was possible between the MCARM header parameters and those required by RLSTAP.

9

A procedure was then developed to utilize the parameter translation table, read the MCARM header parameters and create the VIFF file format for RLSTAP. Using this procedure, it is possible to read MCARM data into RLSTAP for utilization. The procedure was written by the developers of RLSTAP to expedite development of the interface.

4.2.3.2 Perform Interface Testing

A procedure to input MCARM Matlab data files into RLSTAP was tested and MCARM data files were read into RLSTAP for processing. Thus it was demonstrated that RLSTAP could read and process MCARM Matlab data files. The developers of RLSTAP developed the procedure with assistance from Research Associates for Defense Conversion staff. The RLSTAP developers had direct access to the RLSTAP software and had a unique understanding of its operation.

4.2.3.3 Present RL With Interface Alpha Release

An alpha release of the procedure to read and convert MCARM data files for RLSTAP processing was demonstrated and made available to RL. More work is required to encapsulate, enhance, and streamline this procedure. At present, although the procedure does function, it is very convoluted and time consuming.

4.2.3.4 Provide Interface Technical Support

Technical support was provided for the procedure to interface MCARM data with RLSTAP.

4.3 CREST Enhancement and Modification For Bistatic MCARM Data

At the outset, it was decided by RL that the bistatic MCARM data was too sensitive to be made available via the Internet. Hence, this data was not integrated within the MCARM Web pages. However, the structure of the MCARM Web site was designed to accommodate this data when deemed desirable. Support for this data type requires that the following items be satisfied:

- ✓ sufficient disk storage
- ✓ populate the bistatic data repository
- ✓ process bistatic header files
- ✓ change path to point to the right location.

First it will be necessary to free up or acquire disk storage to house the bistatic data. Secondly, the bistatic data must be loaded into the appropriate directory (e.g. httpd/public/bistatic). Then the program 'matlab_hdr' must be executed against each bistatic Matlab formatted data file to create the header files (*.hdr). This program must be executed within the directory where the bistatic data files are to be contained. (Note: the program 'matlab_hdr' must be contained within the search path for executables. The program accepts single or multiple file names and/or

10

wildcards (e.g. *.hdr).) Once this is all completed, the contents of the text file 'env_path' must be modified to point to the appropriate location.

4.3.1 Develop Internet/WWW Procedures

The MCARM Web site was specifically designed so the same procedures can be used to access the monostatic or bistatic data. There is a restriction that only one of the data types can be processed at a time. This is due to the way in which the environmental path variables, CPI_PATH and HDR_PATH, are defined within the file 'env_path' located in the 'htdocs' directory. To process one or the other data type the appropriate path must be specified within this text file to indicate the location of the data (monostatic or bistatic). None of the software needs to be compiled. All of the existing executables will still work. Hence, to search for bistatic data the 'envpath' file would have to be altered to point to the bistatic data. The 'env_path' file can only be altered by the system administrator. Once altered, all users would be permitted to access only that type of data until a subsequent change is made.

The tradeoff is that using this approach the *same* structure, software, and forms can be used for both the monostatic and bistatic data. Hence, for example, the porting to another platform of one data type would be straightforward. However, if there is a need to access both data types concurrently, then the software must be modified to handle multiple paths within the 'env_path' data file.

4.3.2 Make Data Accessible Via Internet

As stated in the previous section, the *same* structure, software, and forms can be used for Internet access of both the monostatic and bistatic data. As currently configured, the MCARM Web site makes no distinction between either data type.

4.4 MCARM Internet Accessibility

The MCARM server can be accessed via the Internet using any available Web browser (e.g. Netscape, Internet Explorer, et cetera). Some display peculiarities may occur depending upon the browser and its version. All of the MCARM data is stored in directories under the ServerRoot (i.e. /xbox/disk0/cavovn/httpd) while the Web pages are stored under the DocumentRoot (i.e. /xbox/disk0/cavovn/httpd/htdocs). Access to the MCARM server begins with the Home Page.

4.4.1 Modify CREST Home Page

Due to the uncertain state of the CREST Home Page during the time of this investigation it was determined that the best course of action was to create a separate MCARM Web Home Page. A more detailed explanation can be found in Section 4.1.

The MCARM Web Home Page is depicted in the figure below. All information and data contained within the MCARM Web site is accessible via this page. Restricted data requires a

user account for access. A user account can be obtained by completing and submitting the appropriate online forms. These forms are accessible from the Home Page.

Navigation through the MCARM Web pages can be achieved by moving the cursor over a text or image hypertext link (cursor arrow turns into a finger upon encountering a hypertext link) and clicking the mouse button. Where appropriate, some pages provide both the text and image links for compatibility with text only browsers. Please see the online User's Guide at URL http://sunrise.deepthought.rl.af.mil/user_guide.html for more information.
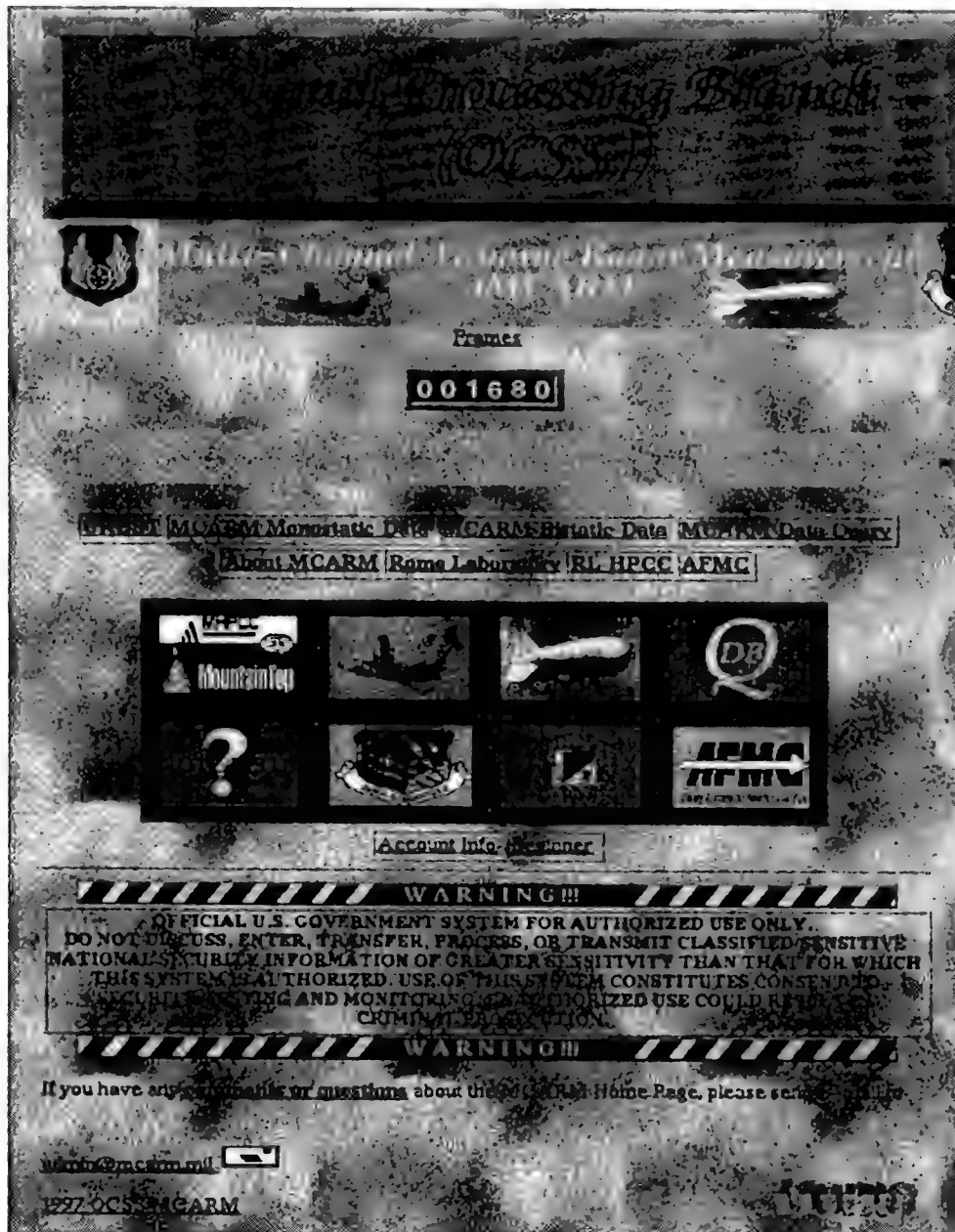
Figure 4.4.1-1 MCARM Home Page

4.4.2 MCARM Data Structure Integration

The MCARM Web hierarchy is depicted in Figure 4.1.1-1. The MCARM data is contained in a 'leaf' of this hierarchy called the Data Library. The Data Library contains both the monostatic and bistatic data. Eventually, the Data Library will be contained within a DBMS. The DBMS will administer and control access to the MCARM data. These functions are presently performed by a system administrator with the aid of software, a Common Gateway Interface (CGI) query.

The MCARM data is housed in two directories located within the directory 'public' contained under ServerRoot. The directories are named 'monostatic' and 'bistatic'. The bistatic directory is presently empty, see Section 4.3. However, both of the directories will have the same structure. Each directory will contain a header file for each data file. (As with the CREST data, the MCARM data is split into separate header and CPI data files.) A data file may reside in that directory or have a link (not to be confused with hyperlink) to the physical location on the system where it resides. The link mechanism permits data files to be scattered across file systems or disk drives when there is insufficient storage available in a single file system. The link mechanism also permits transparent access of data files from a single location via the CGI query software.

4.4.3 CGI Query Interface

CGI C software was written to permit access to the processed MCARM data repository (Data Library). An associated query form is also available. The query interface to the MCARM data is via this form. A user specifies the desired search criteria within the query form. Upon submission, the CGI software processes the request and returns the results of the search to the user. The CGI software is transparent to the user. Also, the CGI software automatically handles concurrent access by multiple users by making use of a user's process identification number.

The CGI C code was designed with 'hooks' for the integration of a DBMS Data Manipulation Language (DML) when the time arrives. This software and the associated query form were specifically designed to provide a more natural user interface and query engine.

Tests were conducted upon completion of the CGI software and its associated HTML form to transfer files via the Internet using a browser. The MCARM WWW Home Page became operational on July 1, 1996 with the successful completion of these tests. The MCARM Web server has been maintained, supported, enhanced and refined since that time. However, the name "www.mcarm.mil" was never registered with the DDN Network Information Center (NIC). This is awaiting local review and clearance. The MCARM Web page can be accessed using the IP address of the host computer or URL "http://sunrise.deepthought.rl.af.mil/".

Subsequently, modifications were made to the CGI query software to incorporation a range capability. This modification permits the retrieval of multiple acquisition numbers instead of just a single acquisition number or data file. This modification required changes to the query form. Also, Mode information was incorporated within the MCARM Web pages for each flight. This information is accessible via the geographic display of each flight path. The selection of a flight

path of interest from the query form will display a new window showing a flight path geographically. From this window it is possible to display a graph of the modes (e.g. LPRF, MPRF, etc.) pertaining to this flight.

4.4.4 Data Dictionary For Variable Translation

RLSTAP and MCARM data files use different names for the radar parameters contained within the files. Hence, variable name translation must occur to use MCARM data within RLSTAP (see Section 4.2.3 for more information).

4.4.5 Perform RLSTAP Needs Assessment

The needs of RLSTAP users regarding the use of MCARM data were carefully assessed. It was determined that real airborne data provided by MCARM would be useful to RLSTAP users. Hence, an investigation was undertaken to determine how the MCARM data could be made accessible to RLSTAP users. To achieve this accessibility goal, three tasks had to be performed: 1) MCARM parameters had to be translated to parameters required by RLSTAP via a translation table; 2) a procedure had to be written to utilize the translation table and read the MCARM header data; and 3) the data had to be converted to the VIFF format (see Section 4.2.3 for more information).

4.4.6 Design, Develop, And Implement RLSTAP/MCARM Interface

All that is required to access MCARM data from within RLSTAP is to have the same parameter names within the header. Hence, once created, a parameter translation table can be used to map MCARM parameters to parameter names used by RLSTAP. This translation table was created to perform the parameter mapping. The translation required the addition of new parameters to the MCARM header and the mapping of existing parameters to those required by RLSTAP. These steps were required because no one to one mapping is possible between the MCARM header parameters and those required by RLSTAP.

A procedure was then written to utilize the parameter translation table, read the MCARM header parameters, and then perform the conversion to VIFF. Using this procedure, it is possible to read MCARM data into RLSTAP for utilization. The procedure was written by the developers of RLSTAP to expedite development of the interface with assistance provided by RADC staff.

4.4.7 Present RL With Initial RLSTAP/MCARM Approach

The RLSTAP/MCARM interface approach was presented to RL personnel for their approval before implementation. In fact, RL personnel were directly involved with the development of the interface from its inception. RL personnel provided invaluable contributions to the design of the interface.

4.5 Provide Technical Support

Technical support was provided during the course of this effort as an integral part of the MCARM system administration. The technical support entailed, among other things,:

- resolution of problems encountered by Internet users (e.g. firewall, file formats, data communication, etc.),
- modification or enhancement of Web pages to facilitate access and ease of use, correct browser page rendering problems, and improve capability and functionality,
- monitoring of server usage (e.g. Perl 'accesswatch' program accessible via the Internet at URL http://sunrise.deepthought.rl.af.mil/usage_log.html or http://sunrise.deepthought.rl.af.mil/daily_log.html, see Figure 4.5-1 below) for resource balancing, security violations, et cetera,
- investigation and correction of HTML problems (e.g. the overlay of flight paths and how they are displayed by different browsers),
- maintained vigilance regarding the future direction of the Internet (e.g. HTML, Java/JavaScript, security, etc.),
- provide support for the use of MCARM data within RLSTAP.
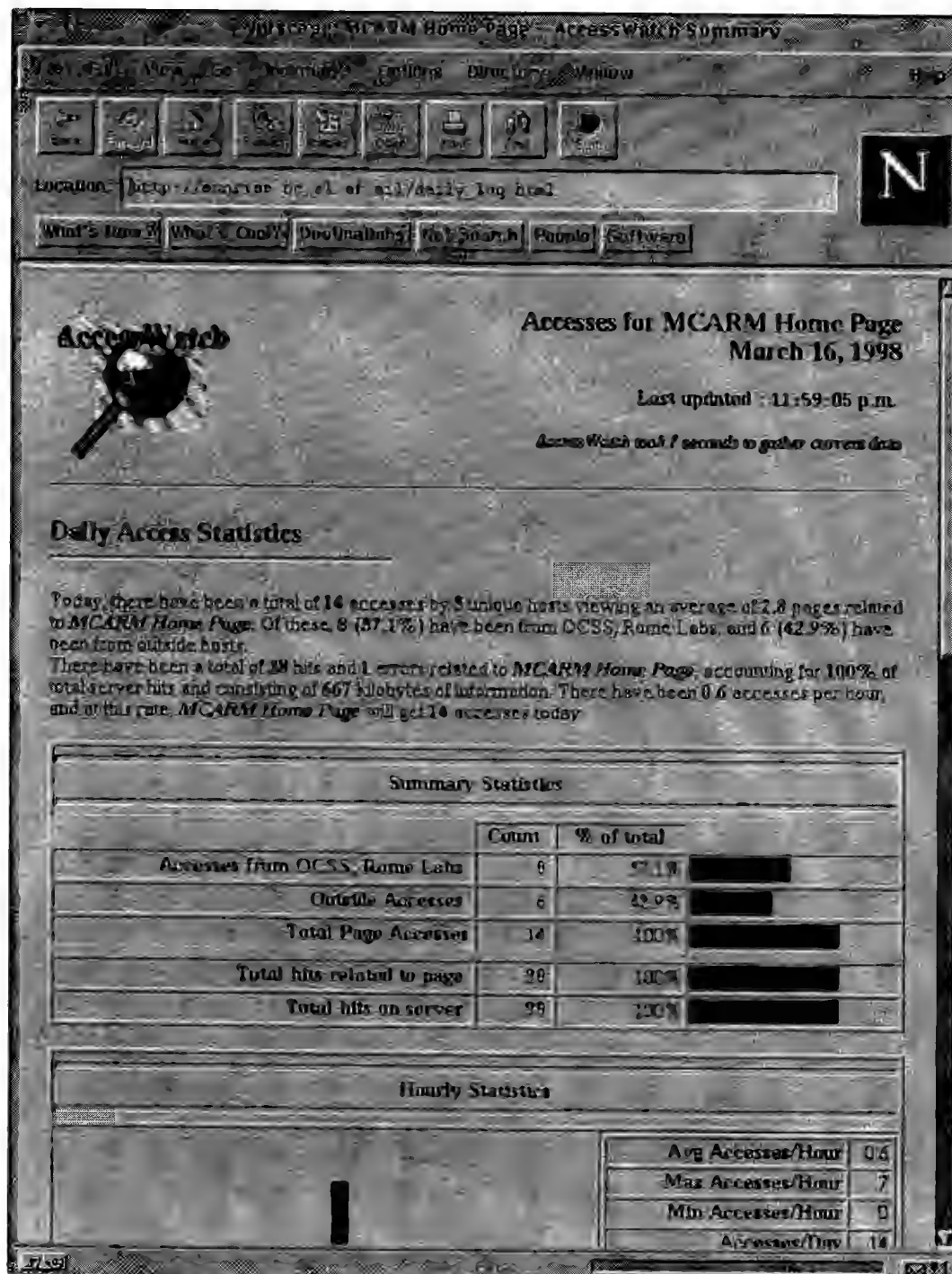
Figure 4.5-1 Daily Access Log

In addition to the above tasks, the computer system upon which the MCARM Web server resides (first 'isaiah' then 'sunrise') was administered. This administration involved the management and upgrade of system resources (e.g. secondary storage, hardware configuration, etc.), and

configuration of the operating system (SunOS 4.1.3) and the associated software (e.g. operating system software patches, X11R5 acquisition and installation, etc.). For example, some system administrative duties were:

- the recompilation of software tools due to upgrades or system migration (e.g. xpaint had to recompiled for 'sunrise' since 'isaiah' was upgraded to the Solaris operating system),
- HPCC Center link had to be modified (port 8001 reference removed),
- establish a new MCARM data file naming convention,
- hardware installation (e.g. installed an 8mm backup unit for 'sunrise' and two 9 gigabytes disk drives for additional MCARM data),
- upgrade of operating system software to correct problems[1] (e.g. Several Sun patches were incorporated to correct a Netscape problem which causes Open Windows to crash. The patches installed made corrections to Open Look and the Unix OS.).

In addition to these functions, it was also necessary to promote MCARM usage by promoting in-house training sessions and participating in work group sessions. For example,

- an in-house MCARM working group meeting was held on December 12, 1996 to promote data access and utilization,
- prepared and presented a talk about the MCARM Internet work at a CREST working group meeting.

Both meetings were well attended. Hands-on assistance was provided for the access and processing of MCARM data in the December 12th meeting.

4.6 Other Relevant Supporting Task

A number of ancillary but supporting tasks were performed during the course of this effort that are not directly referenced in the SOW. These tasks are discussed in this section.

The tasks completed are as follows:

- ✓ investigated the use of Java and Javascript,
- ✓ support of frames,
- ✓ display GPS Latitude/Longitude data points on cartographic reliefs,
- ✓ investigated security issues and mechanisms,
- ✓ developed an Internet User's Guide,
- ✓ developed an Internet Programmer's Guide.

---

[1] Obtained and installed X11 R5 Windows on 'isaiah' and 'sunrise' which was needed for the compilation of the many software tools. These software tools (e.g. 'xpaint') could not be compiled using the current X11 R4 version of Windows. (Could not locate any Sun computers within the Surveillance Facility which had the necessary X11 R5 software libraries. Only 'magnum' came close. The current version of X Windows is X11 R6 but it is not widely used.)

4.6.1 Mapping

The use of Java and JavaScript was investigated for improved functionality and ease of use of the MCARM Internet data. The purpose of this investigation was to circumvent discovered HTML limitations and to stay abreast of Internet trends and future direction.

One critical limitation that needed to be addressed was the display of mapping information from within a browser. A need existed to display flight path information with an associated map relief. Some acquisition files would then be displayed with the associated flight path. This capability permits the visual selection of data files. This method of data selection is more accurate and less haphazard. It was initially thought that Java or JavaScript could be used to circumvent this limitation. After a careful analysis, it was determined that Java would be needed. However, it was discovered that Java was not available for SunOS. Hence, an upgrade to the Solaris operating system would be required before this approach could be pursued further.

JavaScript was implemented in a limited way as a result of the investigation (e.g. lower browser window banner). Also, the study of Java and JavaScript permitted more to be learned about frames. As a result, modifications were made to the MCARM Web pages to support a frame capability to facilitate ease of use.

During this effort, a method was devised to overlay flight path information on a base map. A UTM Delmarva Digital Elevation Model (DEM) formatted map relief was obtained from USGS, via the Internet, for a proof of concept. (Other Eastern Seaboard maps where MCARM flights took place were also located.) Flight path latitude/longitude points had to be translated to the UTM base map projection. Flight number five was tested for overlay anomalies. Although helpful, this method permitted only limited rectification of flight paths, due to warping, with the associated geography. Hence, other methods were investigated to improve the rectification accuracy, such as the concepts of Server/Client Push/Pull and GIS systems. Essentially, the approach was to use these approaches to improve the overlay accuracy.

The Push/Pull approach was quickly abandoned as not feasible. For cost reasons, public domain GIS packages were investigated for their overlay capability. GIS packages can readily create overlays of different map reliefs (e.g. Digital Line Graph (DLG), Digital Elevation Model (DEM), Land Use/Land Cover, et cetera). This overlay capability was attractive. The software package Grass, developed for the Army, was identified and acquired but it proved to be difficult to learn and use. Thus, a search began for other GIS systems.

The investigation focused on identifying COTS mapping packages available and assessing their abilities. ESRI's Arc/Info was selected and evaluated (on paper). Subsequently, a demonstration copy of Arc/Info was found in Building 240 (Mr. Gerald Nethercott's Image Exploitation group). However, due to budget constraints, a copy of Arc/Info could not be acquired and the GIS work proceeded no further.

4.6.2 Internet Security

The initial security mechanisms used involved user accounts and specific computer identification. However, it was strongly felt that this level of security would not be adequate for the MCARM data. Hence, other security issues and mechanisms regarding the access of MCARM data via the Internet were investigated. The focus was on security mechanisms used in secure HTTP. Specifically, these are Secure Socket Layer (SSL), user authentication, and digital signatures. More work on security is required especially if bistatic data is to be made available via the Internet.

The SOCKS software tool was investigated for access of MCARM data through a firewall. SOCKS was investigated because several users were experiencing problems accessing the MCARM data from behind a firewall. The problem was that the IP address of the firewall was being sent instead of the IP address of the computer for which a user was authorized. SOCKS circumvents this problem.

4.6.3 Documentation

It was felt that documentation was needed to assist new and existing users with the access and utilization of the MCARM Internet data. Additionally, detailed information was needed to assist in the administration of the Web site. Consequently, an Internet User's Guide (located at http://sunrise.deepthought.rl.af.mil/user_guide.html) and a Programmer's and System Administrator's Guide (located at http://sunrise.deepthought.rl.af.mil/programmer_guide.html) were developed. For example, the Programmer's and System Administrator's Guide will permit an administrator to perform the following duties:

- Modify and Configure the NCSA MCARM Web server,
- Add new MCARM data files for Internet access,
- Add or modify user accounts,
- Establish data access privileges,
- Maintain and enhance security mechanisms.

4.7 Availability Assessment Of Hosting MCARM Data On The Rome Laboratory HPCC (Option I)

4.7.1 Introduction

The current MCARM data repository resides on a SparcServer 630 MP. The data repository presently consists of 570 preprocessed data files all of which are available via the Internet through the Hyper Text Transfer Protocol (HTTP). A query form and query engine is available to permit a user to search the data repository on any valid MCARM parameter.

The problem is that the data repository is large, about 27 gigabytes, at present, and a data search can take *four* minutes or more for just *one* parameter (e.g. acquisition number). Depending on the system and network load, this time may become even larger. Compounding the problem further is the fact that the search time will definitely increase as the repository grows in size.

Hence, the purpose of this effort was to study, assess, and evaluate the creation of a Multi-Channel Airborne Radar Measurement (MCARM) data base on the High Performance Computer Center's (HPCC) Intel Paragon. Its goal was to significantly improve performance by replicating the Common Research Environment for STAP Technology (CREST) environment, Image Exploitation 2000 (IE 2000) server and data base engine, and the MCARM data repository on the Intel Paragon computer. This effort's primary objective was to create a detailed implementation plan for the performance of this work.

4.7.2 Background

The purpose of this section is to discuss background information that is pertinent to this effort. More specifically, this section will address the Intel Paragon and the MCARM data.

4.7.2.1 Intel Paragon

The Intel Paragon is located in the High Performance Computing Facility at Rome Laboratory in Rome, New York. The facility's mission "...is to support the unclassified RDT&E efforts of all components of the Department of Defense (DoD) by providing interactive remote and local access to hardware, software, and user services with special attention to applications and missions direct input of digitized sensor data."[2] The Intel Paragon may be accessed via the Internet as 'romulus.hpc.rl.af.mil.' The system will soon be connected to the New York NYNET network via a high-speed ATM link. The Intel Paragon operates from behind a firewall.

The Intel Paragon XP/S is a 321-node scalable computer. The compute partition is comprised of 304 MP nodes, each with 64 MB RAM and three i860 processors. Of the 304 nodes, 272 are general compute nodes configured in a 16x17 rectangle, and 32 are equipped with HiPPI daughter cards, each capable of 70 MB/s sustained throughput. Through the HiPPI, the Paragon can take in 2.2 GB/s of sensor data.

The disk subsystem is comprised of eighty gigabytes of disk organized into 20 RAIDs. The disk subsystem is capable of sustaining over 100MB/s into the compute partition. The bulk of the disk space is available for data storage; user file systems are located on a network of workstations. Five Sparc 20 workstations are available to users of the Paragon as development platforms for cross compilation.

---

[2] See URL http://www.rl.af.mil/Tour/HPC/RLHPC.html

20

The Paragon's operating system is a version of UNIX OSF/1 developed by Intel. FORTRAN, C and C++ compilers are provided, including native compilers on the Paragon, and cross-compilers on Sun workstations. The Kuck and Associates signal processing libraries are available for all development languages.

4.7.2.2 MCARM Data

The MCARM program was designed to collect multi-channel clutter data from an airborne platform. The Westinghouse owned BAC 1-11 was used as a platform for the L-Band radar data collection system that was built on a combination of Westinghouse internal funds and MCARM program funds. The system consists of an L-band active aperture antenna, multiple IF receivers, high dynamic range A/Ds, a high speed magnetic tape recorder, a signal/data processor, a multi-mode controller, and operator console. The data was collected at a variety of pulse repetition frequencies (PRFs) over various terrains including mountains, rural, urban, and land/sea interface clutter. Some Westinghouse IR&D testing of Ground Moving Target Indication (GMTI) data were collected with and without foliage to evaluate FOPEN. Mono-static data was collected at PRFs of 7kHz (High PRF), 2kHz (Medium PRF), and 500 Hz (Low PRF). Bistatic data was collected at 23 kHz (High PRF) and at 313 Hz (Low PRF). Westinghouse collected the MCARM data during several Delmarva and East Coast fly-overs that terminated in Florida.

There were 11 flight tests that collected multi-channel data. They are labeled sequentially from 1 to 11. The last two, 10 & 11, were actually done on the same physical flight, but were given different names because flight 10 was the last bistatic flight and flight 11 was a mono-static land/sea interface flight. There was also 2 safety of flight tests that did not collect data, although the radar was tested during the second. These two flights were successful in evaluating the performance and flight envelope of the BAC 1-11 with the external MCARM radome/antenna.

During the MCARM flight test, data was collected simultaneously from a multi-channel sub-aperture architecture and a low side lobe sum and delta analog beam former. The multi-channel architecture was tested using two separate sub-aperture configurations each having 22 degrees of freedom. The analog beam former is useful for comparisons to adaptive beam forming as well as main lobe plus auxiliary configurations. One of the sub-aperture configurations optimized the number of azimuthal degrees of freedom (16) as well as giving up to 2 elevation degrees of freedom. The other configuration was used for all the bistatic data measurement.

Preliminary mono-static data reduction has shown main beam clutter to noise ratios over 80 dB and outstanding adaptive cancellation ratios of up to 70 dB. While preliminary results indicate what appears to be up to 20 dB of residual uncancelled clutter in some of the analyses, good calibration data was collected on a ground range and in the air at the Paxtuent (Pax) River test range. This calibration data showed excellent array performance and can be used to evaluate possible limitations to adaptive cancellation such as channel to channel stability and aircraft effects. This is precisely why this program was necessary as adaptive sensors are emerging to meet the challenges of future air surveillance threats. The ground range data showed up to -58 dB RMS side lobe performance for an analog beam former and localized closed loop adaptive nulls of -90 dB with respect to peak. This data was augmented by airborne pattern

measurements, which showed -47 dB RMS side lobes for the analog beam former. These measurements show the degradation due to aircraft effects that STAP should be able to overcome.

4.7.2.2.1 MCARM DATABASE

The MCARM database is comprised of preprocessed airborne radar datacubes. Each acquisition data file consists of a single coherent processing interval (CPI). The data files have undergone the following preprocessing steps: baseband conversion, decimation, digital in-phase and quadrature phase (I/Q) conversion, pulse compression, and channel balancing. Each preprocessed datacube, and its associated "header" variables, are provided in MATLAB format. Header variables include transmit pointing direction, number of interpulse periods, pulse repetition frequency, aircraft platform inertial navigation data, global positioning system (GPS) data, etc. In addition, header variables have been added (e.g. PRF) to facilitate processing and maintain compatibility with CREST and RLSTAP. A complete description of these variables can be found in Appendix A.

The original 'raw' data files, before the preprocessing step, are approximately 15 megabytes in size. These data files have been archived onto rewriteable optical storage media. The processed data files are contained on hard disk drives and occupy approximately 27 gigabytes of storage. The data files double in size after the preprocessing step is performed. Approximately, 55 gigabytes of storage is required to contain all of the processed 'raw' data files.

The preprocessed multidimensional measured radar data is stored in the variable named "CPI1", referred to as the datacube. The format for this datacube is contained in the datacube figure below. At the present time, the database contains 561 Matlab formatted data files.

22
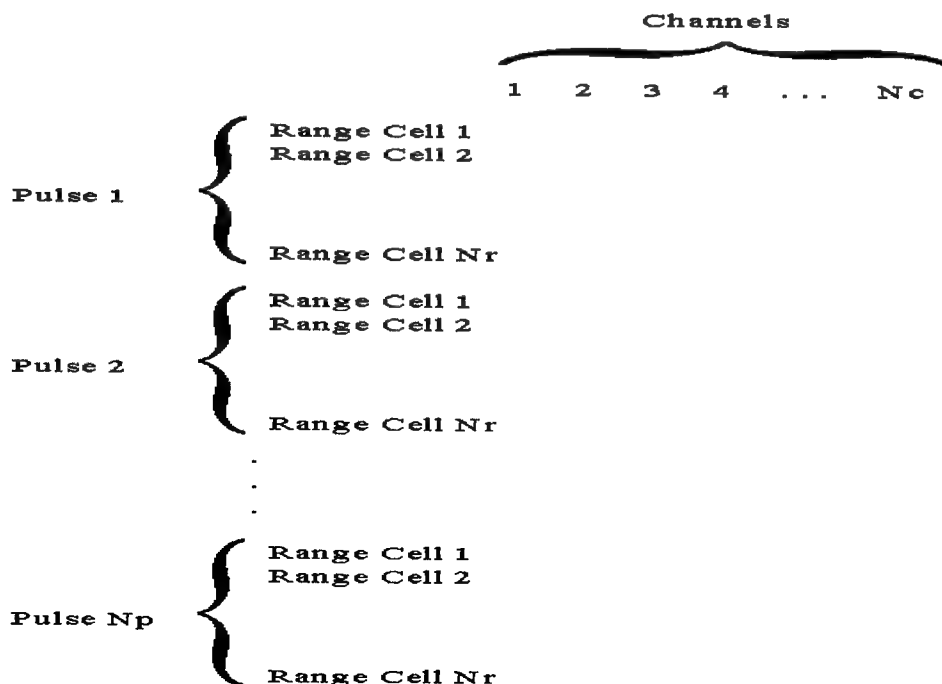
4.7.3 MCARM Data HPCC Repository Assessment

The CREST/RLSTAP ADT tool has taken many years to develop. At the time of this writing, a new contract has been awarded to incorporate new and advanced technology within this environment. As such, the environment is very dynamic, fluid and complex. The interfacing of new user data, as the MCARM data, is not often easy or straightforward. For example, the MCARM data must first be converted to the internal VIFF format and the header parameters properly mapped before the data can be utilized within the RLSTAP/ADT tool. Additionally, some researchers may resist the use of the RLSTAP/ADT processing paradigm. These researchers may instead choose procedures and mechanisms that they may be more accustomed and familiar with, such as the use of MCARM data in conjunction with Matlab.

The focus of the Intel Paragon rehosting of the MCARM data was the replication of the CREST and IE 2000 environments. The goal of the rehosting is to replicate the Internet functionality currently available on the 630MP SparcServer on the Paragon. Its aim is to borrow the client-server concepts of the IE 2000 for the implementation of the CREST Data Library and its architecture. Following the performance of a thorough investigation of CREST and the IE 20000 it was felt that the inherent client/server nature of the Internet was adequate for the dissemination of the MCARM data, without the introduction of additional complexity. Hence, it was determined that the Paragon Internet MCARM implementation could make use of basic concepts provided by the Web.

4.7.4 Implementation Plan

A number of options were considered for the implementation of the MCARM data repository on the Intel Paragon, namely:

- Host the data repository on the Paragon
- Utilize the data processing ability of the Paragon.

An initial assessment indicated that the first option was not feasible. First, the Paragon could not afford to set aside the amount of disk storage that would be required to house the MCARM data repository. (The Paragon has about 80 gigabytes of disk storage. About 54 gigabytes would be required for the MCARM data repository.) Secondly, there would be a significant increase in the network traffic to the Paragon, assuming that the data repository would not be hosted there. Thirdly, access to the MCARM data repository, through the Internet, would be more difficult because the Paragon lies behind a firewall. Taking all of these factors into consideration it was deemed more feasible to implement the second option.

As it is presently configured, the MCARM data repository is split into distinct header and data files due to the size of the processed data (about 30 megabytes). A query of the data repository, using the Common Gateway Interface (CGI) C-based query engine, uses only the header files for query qualification. The header files are searched to identify data files that meet the search criteria specified by the user. Once the appropriate data files have been identified a simple file

naming association is used to identify the appropriate data files. Thus, any query of the MCARM data repository requires that the data files be searched for the established criteria.

Utilizing the Paragon for its rapid data processing ability offered several tradeoffs. The header files could be hosted on the Paragon or transmitted to the Paragon whenever a query of the repository was performed. The former case had the disadvantage that the appropriate header files would have to be updated whenever the MCARM data repository was updated but it significantly reduced the network traffic. In the latter case, the header files would have to be transmitted to the Paragon whenever a query was performed. This would adversely affect the network bandwidth and slow performance. The header files presently require about 75 megabytes of storage; a number that could easily double. The transfer alone could take several minutes and consume a significant portion of the communication bandwidth.

After careful consideration, it was decided that the best approach would be to host only the header files on the Paragon. Using this approach, the search criteria would be transmitted from the SparcServer to the Paragon for qualification. The Paragon would then return the results of the query to the SparcServer. The communication between the SparcServer and the Paragon would be implemented using sockets. In this case, only the query results would have to be transmitted back to the SparcServer. The worst case scenario, which should be extremely rare, is that the entire data repository qualifies the search. Several minutes may be required to transmit the results back in this case. Otherwise, the results of the query should be available in a matter of a few seconds. In addition, this approach would not incur any additional cost. *No* hard disk drives would have to be acquired to host the MCARM data repository on the Paragon.

4.7.4.1 Implementation Tasks and Schedule

The following diagram provides a summary of the implementation tasks and the anticipated scheduling for the HPCC Intel Paragon implementation.

Tasks	1	2	3	4	5	6
Rehost Header Files	◄---►					
Develop Sparc/Paragon Sockets Interface	◄------------------------►					
Software Testing					◄--------►	
Interim Report						◄---►

Key:
     ^ - Monthly Status Reports

Development of the Sparc/Paragon sockets interface would entail the modification of the CGI software contained in Appendix B.

24

5.0 Recommendations/Conclusion

Within this report RADC discussed the MCARM Web server, the MCARM data repository, and its access via the Internet. This report also discussed the implemented data structure and the established Internet access mechanisms. Also included within this report is the documentation needed to utilize and maintain the MCARM data repository and Internet Web site.

Following is a list of items that should be considered regarding the future direction of the MCARM Web server. The objectives of these items are to improve functionality and to maintain the state-of-the-art of the MCARM Web server:

- Operating System Upgrade

  The current operating system, SunOS 4.1.3, is no longer supported. Software tools needed to maintain and enhance the MCARM Web server are difficult to find or non-existent (e.g. Java, VRML, 3-D, etc.). Hence, a migration to the Solaris 2.x operating system should be seriously considered. The migration will have the following benefits:

  - ✓ supported operating system with a future migration path
  - ✓ portable and standards compliant OS (Solaris is based on System V Release 4 (SVR4) of the Unix OS. Solaris runs on platforms ranging from PCs to supercomputers)
  - ✓ improved functionality
  - ✓ availability of more software tools (public domain and COTS)
  - ✓ availability of Java (this is where the industry is going for cross-platform compatibility and improved Web pages).

- Acquisition and Implementation of Java/VRML

  This is clearly where the Internet is headed. Java will eventually replace HTML with respect to Web page presentation and cross-platform compatibility. Java permits Web pages to be more dynamic, engaging and interactive. Java should obviate the need for the Common Object Request Broker Architecture (CORBA), being used or promoted by CREST, for client-server applications in a heterogeneous distributed environment.

- Investigate MCARM Data File Size (compression/decompression)

  Determine what can be done to reduce the size (approx. 30MB) of the processed Matlab data files. The file size problem is due to the addition of imaginary data and the way Matlab handles this data type when the original 'raw' files are loaded for processing. The size of these files makes transmission difficult and time consuming. One possibility may be the transmission of algorithms, with the associated data, which may then be processed automatically (e.g. via Java) on the target computer. This would permit better compression of the source data and expansion of the file size on the target computer.

25

- Acquisition and Implementation of Mapping Software

  We have flight paths that cover most of the U.S. Eastern Seaboard. A need exists to display the flight paths over the appropriate geographic areas for more accurate visual selection of acquisition data. Also, a COTS GIS package permits a number of overlays to coexist. A number of maps types are available from a number of sources, for free, but we lack the software tool needed to construct the proper map reliefs showing elevation, roads, land usage and cover, et cetera. The public domain tool GRASS (developed by the Army) is inexpensive but hard to learn and difficult to use. In addition, it is being phased out, is no longer supported and is being replaced by COTS tools.

- Implement Better Security Mechanisms

  The initial security mechanisms used involved user accounts and specific computer identification. However, it is strongly felt that this level of security will not be adequate for the MCARM data in the future. This is especially true if bistatic data is to be made available via the Internet. Hence, other security issues and mechanisms regarding the access of MCARM data via the Internet should be investigated. Specifically, these are Secure Socket Layer (SSL), user authentication, and digital signatures.

- More work is required to encapsulate, enhance, and streamline the MCARM/RLSTAP procedure. At present, although the procedure does function, it is very convoluted and time consuming. The development of interface software and the integration of this software into RLSTAP will significantly improve this interface.

Also, within this report, RADC has provided recommendations for the implementation of the MCARM data repository on the Rome Laboratory HPCC or Intel Paragon. After a careful and thorough analysis it has been determined that the best alternative available is to develop a communication link between the present Internet MCARM data server (SparcServer 630MP) and the Paragon computer. The communication link would be established through the use of socket communication. This approach affords: the most cost-effective and efficient utilization of the subject hardware; it would not incur additional cost for disk drive hardware; and does not raise security concerns. In this approach, the Paragon would function as a 'back-end processor' to accelerate the database search process. This approach does, however, raise communication bandwidth concerns which require further investigation.

Appendix A – MCARM Data Variables Description

Each acquisition contains the following MATLAB variables listed in alphabetical order. (See the MATLAB Reference Guide for information about how to load MATLAB formatted files into the MATLAB environment.) Variables marked n.a. (not applicable) can be ignored since they apply to other modes of the Westinghouse radar that were not used for data acquisition.

AcquisitionNumber	This integer should agree with the acquisition number portion of the filename.
ApplyRxMatchOn	n.a.
AttenChannels	These floating point values are the receiver attenuation settings for each of the MCARM channels for the beam position
BeamSteeringType	This integer in conjunction with RcvrMode determines the mode of the radar.
BlockSize	n.a.
CPI1	Baseband, Digital I/Q, Pulse Compressed, Channel balanced samples of multi-channel coherent processing interval (CPI) radar data. This data is in the form of an S row by M column matrix of integer values, where S is the number of samples per CPI and M is the number of receiver channels. For MCARM data, S equals RangeCellsPerIPP * IppsPerCpi For MCARM data, M is always 24. Channels (columns) 1 and 9 are the outputs of the sum and delta manifolds respectively. The other 22 channels are array subaperature outputs that depend on which array configuration was used. The sample values integers representing 12 bit, signed A/D converter samples with two additional bits added as the least significant bits. Bit 0 is the range zero bit that is 1 only during the first A/D sample of each IPP. Bit 1 is the A/D saturation bit that is 1 only if the input is outside of the A/D input range.
CfarThreshold1	n.a.
CfarThreshold2	n.a.
CfarThreshold3	n.a.
ChannelMatchCoeffs	n.a.
ChannelSumOn	n.a.
ChannelPerCPI	For MCARM data this value is always 24
CheckROs	n.a.
ClutterFrequency	For MCARM data this value is normally 0, however, it represents a frequency shift in hertz applied to the transmit pulse. The purpose of this is to allow the

radar to move the doppler frequency of the main lobe clutter to zero.

ClutterTestOn	n.a.
ComputeRxMatchOn	n.a.
CpisPerAquisition	This integer is the number of CPI data acquisitions included in the file. Currently only one CPI is recorded per file and is stored in variable CPI1. Future enhancements will allow multiple CPIs, with different PRFs, to be recorded. In that case variables CPI1, CPI2, etc. will be stored in the data file. When there are multiple CPIs, associated header variable values will be represented as vectors (e.g. IppsPerCpi(1) for CPI1, IppsPerCpi(2) for CPI2, ...IppsPerCpi(n) for CPIn)
DiqDecimation	This integer indicates the amount of oversampling that exists in the data relative to a 0.8 usec range cell. Typically this value will be 4, indicating that data was sampled at 0.2 usec intervals and can be decimated 4 to 1 to obtain one sample per range cell.
DiqOn	A value of zero indicates that the data is real A/D sample data. A value of 1 indicates that I/Q processing was performed prior to writing this file.
DisplavChannelSelec	n.a.
DisplayIppOrRgSelec	n.a.
DisplaySelection	n.a.
DolphdB	n.a.
EnvelopeDetectSelec	n.a.
FlightNumber	This variable indicates the flight number.
FTAAHARSCompassRead	n.a.
FTAAHARSPitch	n.a.
FTAAHARSRoll	n.a.
FTAAHARSValidity	n.a.
FTABAPressureAlt	Target plane barometric pressure altitude in feet.
FTADataValid	Indicates the validity of target plane navigational data. A value of one indicates that the data is valid. A value of zero indicates that the data is invalid or that navigation data is not being received.
FTAGPSGroundSpeed	Target plane ground speed in knots calculated from GPS data.
FTAGPSGroundTrack	Target plane ground speed true North heading in 0.1
FTAGPSLatitude	Target plane latitude or Rome Laboratory Multiple Target Simulator (MTS) latitude in degrees.
FTAGPSLongitude	Target plane longitude or Rome Laboratory Multiple Target Simulator (MTS) longitude in degrees.
FTAGPSTime	Target plane current time from GPS data in milliseconds from midnight GMT.

FTAGPSAltitude	Target plane altitude from GPS data in feet above mean sea level.
FTARAHeight	Target plane radar altimeter reading in feet, but was not in use during MCARM data acquisition
FftOn	For MCARM data, this value is normally zero. If nonzero, it indicates that the radar performed Doppler processing of the data.
FftWeightSelect	For MCARM data, this value is normally zero. If nonzero, it indicates that the radar applied weighting to the data prior to Doppler processing.
FilterIncrement	n.a.
FilterRangeGate	n.a.
FilterStart	n.a.
FilterStop	n.a.
GPSAltitude	MCARM platform altitude from GPS data in feet above mean sea level.
GPSLatitude	MCARM platform latitude from GPS data in degrees.
GPSLongitude	MCARM platform longitude from GPS data in degrees.
GPSNavDataValid	Binary coded status from GPS data. A value of 8192 (2^{13}) indicates valid data. Other values indicate GPS errors.
GPSSystemTime	MCARM platform current time from GPS data in seconds from midnight GMT.
GPSTimeOfXmit	Value is normally zero. If used, it is an (16-bit) integer representing the time that the GPS data was transmitted measured in 64 microsecond periods since system initialization.
GPSTimeValid	A value of zero indicates that the GPSSystemTime value is valid.
GPSVelocityEast	The east component of MCARM platform from GPS data in knots.
GPSVelocityNorth	The north component MCARM platform velocity from GPS data in knots.
GPSVelocityVert	The vertical component MCARM platform velocity from GPS data in knots where positive indicates upward direction.
GatesToDump	n.a.
GatesToRecord	n.a.
GuardThreshold1	n.a.
GuardThreshold2	n.a.
HoleSize	n.a.
INUAltitude	MCARM platform altitude in feet from INU data. The altitude is calculated from inertial navigation data and subject to long term error.
INUAltitudeDelta	This integer is the time when attitude data was sampled by the INU relative to the time when velocity data was

	sampled by the INU measured in 14.75 microsecond periods. The time when the velocity data was sampled is recorded in INUVelocityTime.
INUDataValid	A value of zero indicates that INU data is valid. A nonzero value indicates invalid INU data.
INUDriftAngle	The angle between the MCARM platform center line and the ground velocity direction in radians from INU data.
INUGroundSpeed	MCARM platform ground speed in feet per second from INU data.
INULatitude	MCARM platform latitude in radians from INU data. Positive angles are north of the equator.
INULongitude	MCARM platform longitude in degrees from INU data. Negative angles are west of the prime meridian.
INUPitch	MCARM platform pitch in radians from INU data. Positive angles are nose up.
INUPlatformAzimuth	This value is the angle between the navigation system's inertial x-axis and the MCARM platform center line in radians from INU data.
INUPlatformRoll	MCARM platform roll angle in radians from INU data. Positive angles are right wing down.
INUVelocityTime	This integer is the number of 64 usec periods since initialization of MCARM platform INU system.
INUVelocityX	MCARM platform velocity along the navigation system's inertial x-axis in feet per second from INU data.
INUVelocityY	MCARM platform velocity along the navigation system's inertial y-axis in feet per second from INU data.
INUVelocityZ	MCARM platform velocity along the navigation system's inertial z-axis in feet per second from INU data. The z-axis is vertical relative to the center of the earth with positive values up.
INUWanderAngle	This value is the angle between the navigation system's inertial x-axis and true north in radians from INU data.
IppsPerCpi	This integer value is the number of pulses (inter-pulse periods) that occurred during this acquisition. The number of pulses recorded is IppsPerCpi NumSpaceChargePulse.
ManifoldSwitch	Normally 0. A value of zero indicates that channel 1 is switched to the test manifold during the transmit pulse. A value of 1 indicates that channel 1 is switched to an auxiliary input during the transmit pulse.
McarmMesaMode	This integer value will be zero for all MCARM data.
MesaOptions	n.a.
MesaProcessing	n.a.
ModeName	This is a MATLAB string variable containing the name assigned to the mode of the MCARM radar when this data

	was recorded.
ModuleMapChannel	This variable is a 1 by 24 vector of binary encoded integers. Each element of the vector corresponds to a MCARM receiver channel. Element ModuleMapChannel(i) Indicates which of the 32 T/R modules in the MCARM array are connected to receiver channel i. Each element of ModuleMapChannel should be treated as a 32 bit 2's complement integer where each bit corresponds to a T/R module. The MSB corresponds to module 31 and the LSB corresponds to module 0. A bit value of one indicates that the corresponding module is connected and zero indicates that it is not connected. Typically elements 1 and 9 have the value -1, i.e., all bits are 1. This is because channels 1 and 9 are the sum and difference manifolds and are connected to all modules.
NoiseLevelTestOn	n.a.
NumRangeGateZero	This integer represents the number of range cells (0.8 usec) during an inter-pulse period (IPP) during which the receivers are not being recorded. When this value is zero, the receivers are being recorded during the entire inter-pulse period, including during the transmit pulse.
NumSpaceChargePulse	This integer value represents the number of inter-pulse periods at the beginning of a coherent processing interval (CPI) that were not recorded. The intent of this feature is to delay recording so that all range ambiguous echo pulses are being recorded. The number of pulses recorded is IppsPerCpi - NumSpaceChargePulse.
Passthru	n.a.
PcWeightSelect	Normally 0. If nonzero, it indicates that the radar applied weighting to the data prior to pulse compression.
PlaybackControl	n.a.
PrfCombineOn	.a.
Prf	Pulse repetition frequency.
PulseCompressSelect	A value of one indicates that I/Q processing was performed prior to writing this file.
PulseToPulseSelect	n.a.
PulseWidth	This value is the width of the transmit pulse, which is 0.8 usec per range cells.
QuickLookOn	n.a.
RangeCellsPerIPP	This integer is the number of range cells recorded during this CPI.
RangeGateFilter	n.a.
RangeGateIncrement	n.a.
RangeGateSelectOn	n.a.

RangeGateStart	n.a.
RangeGateStop	n.a.
Rawdata	Indicates which data file is being processed
RcvArrayWeight	n.a.
RcvAzBWShape	This integer indicates which array tuning file was used for receive beam shape and steering.
RcvAzPointAngle	n.a.
RcvAzPointIndex	This integer is the beam position index selected for the receive mode.
RcvElPointAngle	n.a. (Elevation steering is determined by azimuth steering.)
RcvElPointIndex	n.a. (Elevation steering is determined by azimuth steering.)
RcvError	n.a.
RcvFrequency	The receiver center frequency used during receive mode in MHz.
RcvrMode	This integer in conjunction with BeamSteeringType determines the mode of the radar.
RecordOn	n.a.
Rxtest	Indicates which test file was used for channel balancing
RxExpectedWeight	n.a.
RxSensitivityTstOn	n.a.
RxStabTestOn	n.a.
StcOn	n.a.
SteerArray	Steering vectors to be used for Flights 5-11
TxExpectedWeight	n.a.
TxPowerTestOn	n.a.
TxStabTestOn	n.a.
WESTINGHOUSE	This is a MATLAB string variable containing a postflight entered comment describing the data acquisition in the file.
WaveForm	This integer indicates the transmit waverforn type. A value of 0 indicates continuous wave transmission, 1 indicates 14.4 usec LFM transmit pulse, 2 indicates 50.4 usec LFM transmit pulse, 3 indicates 100 usec LFM transmit pulse.
XmitArrayWeight	n.a.
XmitAzBWShape	This integer indicates which array tuning file was used for transmit beam shape and steering.
XmitAzPointAngle	n.a.
XmitAzPointindex	This integer is the beam position index selected for the transmit mode.
XmitElPointAngle	n.a. (Elevation steering is determined by azimuth steering.)
XmitElPointIndex	n.a. (Elevation steering is determined by azimuth

XmitFrequency   steering.)

XmitFrequency   The transmitter center frequency used during transmit mode in MHz.

Notes:

1. Inertial Navigation Unit (INU) data is sampled before and after each acquisition. For this reason, each variable is a 1 by 2 vector. Time delay between samples can be calculated from INUVelocityTime as follows:

$$(\text{INUVelocityTime}(1,2) - \text{INUVelocityTime}(1, 1)) * 64 \text{ usec}$$

2. When the target plane navigational data is not being recorded and the Rome Laboratory Multiple Target Simulator (MTS) is being received, the MCARM radar inserts the latitude and longitude coordinates of the MTS in place of the target plane latitude and longitude.

3. Geographic boresight angle of the MCARM radar, i.e., relative to true north, is calculated as follows:

$$(\text{INUPlatformAzimuth} - \text{INUWanderAngle}) * 180/\text{pi} - 90 \quad (\text{degrees})$$

where east is +90 degrees.

33

Appendix B – Common Gateway Interface (CGI) Source Code

```
/*
#
#
#
#    MCARM Inquiry Form
#
#
# created 12/5/95 author: V.N. Cavo
# last modified on 7/24/95
#
# 6/18/96 - Modified software to display the Matlab compressed file (.gz) instead of the just
#       the CPI suffixed file. The compressed file contains the header information. The
#       environment variable for CPI_PATH was not changed. It now points to the location
#       of the compressed Matlab files.
#
# 7/24/96 - Modified software to incorporate value range capability. Also, corrected a bug
#       where an erroneous error was received when no match was found.
#
# 11/20/96 - Took message blink out and updated below message.
#
# Note: read_hdr.c routine must be compiled with query_form.c
#     cc query_form_new.c read_hdr_new.c -o query_form_new
#     The query_form executable must then be placed in 'cgi-bin' where
#     it is invoked by 'query_new.html' upon query submission.
*/
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <ctype.h>

#define PERMS 0666
#define STRLEN 5000     /* Define stdin buffer size parameter */
#define MAX_PAIRS 1000   /* Define number of name/value pairs parameter */
#define PATHLEN 100     /* Define environment variable HDR_PATH length */
#define TEXT_SIZE 500   /* Concatenated file name string */
#define MODULO_OP 4     /* Number of files per line to display */

struct pairs {
        char *name;             /* Parameter name */
        char *value;            /* Parameter value */
        short int condition;    /* Criteria True(=1) or False(=0) condition */
} pairbuf[MAX_PAIRS], search[MAX_PAIRS];   /* allow for MAX_PAIRS name/value
pairs */
```

```c
struct triples {
        char *name;             /* Parameter name */
        char *from;             /* Parameter from value */
        char *to;               /* Parameter to value */
        short int condition;    /* Criteria True(=1) or False(=0) condition */
}  criteria[MAX_PAIRS];         /* allow for MAX_PAIRS name/from/to triplets */

int pairidx=0, searchidx=-1, getpid(), no_rec_found=0, none_flag=0;
char cmd[MAX_PAIRS], list_name[PATHLEN]; /* System call command string */
FILE *fopen(), *fd, *fd_env, *fd_cpi;
extern short int read_hdr();


main(argc, argv)
int argc;
char *argv[];
{

char
buffer[STRLEN],*ptr,*ctime(),datebuf[26],line[PATHLEN],buff[PATHLEN],*fgets(),hdr_p
ath[PATHLEN],cpi_path[PATHLEN];
register int len, clen, i, j, startptr, counter;
long clock, time();
unsigned short boolean=1; /* Set logical operator 1 (AND) 0 (OR) */

        /* get system date and time for timestamp */
        clock=0;
        time(&clock);
        ptr = ctime(&clock);
        strncpy(datebuf,ptr,26);
        printf("%s\n",datebuf);

        /* Open the output file */
        if((fd = fopen("/tmp/query_search","w")) == '\0') {
                fprintf(fd,"main: Cannot open temporary file '/tmp/query_search");
                exit();
        }
        /* Open the environment file and get HDR_PATH and CPI_PATH */
        /* Could not set HDR_PATH and CPI_PATH in CGI environment. Hence, next best
thing was
        to read these environment variables from a file to preclude recompilation.
        Note: CGI environment is different then user environment (i.e. env > /tmp/env_path
        puts CGI env into target file. That is, CGI sends its own environment to server. */
        if((fd_env = fopen("/xbox/disk0/cavovn/httpd/htdocs/env_path","r")) == '\0') {
                fprintf(fd,"main: Cannot open temporary file
'/xbox/disk0/cavovn/httpd/htdocs/env_path");
                exit();
```

35

```
}
/* Set the Header and CPI path environment variables */
fgets(line,PATHLEN,fd_env);
strcpy(hdr_path,(strpbrk(line,"=")+1));
for(i=0;hdr_path[i] != NULL;i++)  /* Get rid of newline character */
        if(hdr_path[i] == '\n')
                hdr_path[i] = '\0'; /* Replace newline with NULL character */
fprintf(fd,"HDR_PATH = %s\n",hdr_path);
fgets(line,PATHLEN,fd_env);
strcpy(cpi_path,(strpbrk(line,"=")+1));
for(i=0;cpi_path[i] != NULL;i++)  /* Get rid of newline character */
        if(cpi_path[i] == '\n')
                cpi_path[i] = '\0'; /* Replace newline with NULL character */
fprintf(fd,"CPI_PATH = %s\n",cpi_path);
fclose(fd_env); /* Close the environment file */
/* Create the header and CPI directory links */
/*      sprintf(cmd,"unalias rm;cd /xbox/disk0/cavovn/public_html;rm -f cpi hdr",hdr_path);
/* delete Header/cpi path link */
        sprintf(cmd,"unalias rm;cd /xbox/disk0/cavovn/public_html;rm -f cpi hdr"); /* delete
Header/cpi path link */
        system(cmd);
        sprintf(cmd,"cd /xbox/disk0/cavovn/public_html;ln -s %s hdr",hdr_path); /* Header
path link */
        system(cmd);
        sprintf(cmd,"cd /xbox/disk0/cavovn/public_html;ln -s %s cpi",cpi_path); /* CPI path
link */
        system(cmd);


/*
        Process environment variables
*/
/* process POST(recommended) method only */
if(strcmp(getenv("REQUEST_METHOD"),"POST")) exit();
/* ignore url encode forms */
/* Output of a form being processed? */
if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
exit();
/*      Some variables and constructs are described as being `URL-encoded'. In a URL
        encoded string an escape sequence consists of a percent character ("%")
        followed by two hexadecimal digits, where the two hexadecimal digits form
        an octet. An escape sequence represents the graphic character which has the
        octet as its code within the US-ASCII coded character set, if it exists. If
        no such graphic character exists, then the escape sequence represents the
        octet value itself.
*/
```

```c
/*      strcpy(line,getenv("HTTP_ACCEPT"));
        printf("HTTP_ACCEPT = %s\n",line);*/


        /* get stdin length from environment variable */
#ifdef OLD /* support for non-CGI server (e.g. NCSA 1.0a5 or earlier) which used
            command-line arguments */
        clen = atoi(argv[1]);
#else   /* get string from environment variable */
        clen = atoi(getenv("CONTENT_LENGTH"));
#endif

        if(clen > STRLEN) {
                printf("Buffer overrun\n");
                exit();
        }
        /*
                Output the HTML information
        */
        /* print the CGI required header information */
        /* specify full document and MIME type - Note: blank line required */
/*      printf("Content-type: text/html\n\n");*/
/*      printf("Content-type: application/x-gzip\n\n");*/
        /* print document title and initial heading */
        printf("<HTML>\n");
        printf("<HEAD><TITLE>MCARM Inquiry Form Response</TITLE></HEAD>\n");
        printf("<BODY><H1><IMG
SRC=\"/icons/mcarm_small.gif\">MCARM<BR>Inquiry Response</H1>\n");
        gets(buffer);  /* get the stdin input user response buffer - from HTML Form */
        /* Bug - strlen function reports 1 more character than reported by environment
variable
                CONTENT_LENGTH. Unexpected results occur if value returned by strlen
used!
                Work around is to use value reported by CONTENT_LENGTH. The function
strlen may be
                counting the terminating NULL character, contrary to documentation. */
        len = strlen(buffer);
/*      if(len != clen) printf("len=%d, clen=%d\n",len,clen); */
/*      printf("%s\nlen=%d\n",buffer,len);  */
        buffer[clen] = '&';  /* Terminate last name/value pair */
        buffer[len] = '\0'; /* Terminate buffer string */
        /* Replace '+' and hexadecimal special characters */
        for(i=0;i<len;i++) {
                if(buffer[i] == '+') buffer[i] = ' ';
                if(buffer[i] == '%') { /* hex special character? */
                        buffer[i] = '?'; /* prepare for packing */
                        buffer[i+1] = '?';
```

```
                            buffer[i+2] = '?';
            }
}
/* Pack the buffer to remove redundant blanks */
pack(buffer, len);
/*                          · */
/* Split the buffer into name/value pairs */
/*                          */
len = strlen(buffer);  /* Get new string length */
startptr = 0;
for(i=0;i<len;i++) {
            if(buffer[i] == '&') {
                        buffer[i] = '\0'; /* set ampersand to null */
                        split(buffer,startptr,i);
                        startptr = i+1;
                        if((++pairidx) > MAX_PAIRS) break;
            }
}
/*                  */
/*                  */
/* Process the form information */
/*                  */
/*                  */
/* Print timestamp and header information */
fprintf(fd,"%s\n\n", datebuf);
/* Process user supplied information and perform some error checking */
for(i=0;i<pairidx;i++) {
            for(j=0;pairbuf[i].value[j] != '\0';j++) /* Convert all characters to uppercase */
                if(islower(pairbuf[i].value[j]))
                        pairbuf[i].value[j] = toupper(pairbuf[i].value[j]);
            if(((i % 3) == 0)) /* Parameter NONE specified during parameter name/value
pair cycle? */
                        if((!strcmp(pairbuf[i].value, "NONE")))
                                none_flag = 1;
                        else
                                none_flag = 0;
            /* Push the parameter value onto data base stack - place holder for DBMS */
            if(!strcmp(pairbuf[i].value, "AND")) /* Set boolean to AND */
                        boolean = 1;
            else if(!strcmp(pairbuf[i].value, "OR")) /* Set boolean to OR */
                        boolean = 0;
            /* Save parameters if not 'NONE' or NULL */
            else if(strcmp(pairbuf[i].value, "NONE") && strcmp(pairbuf[i].value, "\0")) {
                        if(none_flag) {
                        /* Some parameter 'NONE' set to a value */
```

38

```c
                        printf("<H3><BLINK><I>ERROR - PARAMETER 'NONE'
SET TO VALUE</I></BLINK></H3>\n");
                        printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                        break;
                    }
                    else {
                        search[++searchidx].name = pairbuf[i].name;
                        search[searchidx].value = pairbuf[i].value;
/*                      printf("Name = %s, Value = %s\n", search[searchidx].name,
search[searchidx].value);*/
                    }
                }
                /* Parameter FROM value empty? */
                else if(((i % 3) == 1) && (none_flag == 0) && !strcmp(pairbuf[i].value,
"\0")) {
                        printf("<H3><BLINK><I>ERROR - PARAMETER 'FROM' VALUE
EMPTY</I></BLINK></H3>\n");
                        printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                        break;
                }
            }
            /*                      */
            /*                      */
            /* Develop the search criteria  */
            /*                      */
            /*                      */
            fprintf(fd,"Boolean = %d\n", boolean);
            if(searchidx >= 1) {
                fprintf(fd,"There are %d search conditions to satisfy\n\n", ((searchidx +1)/2));
                for(i=0;i<=searchidx;i++) {  /* Output the non NONE values used for DB
search */
                        fprintf(fd,"%s = %s\n", search[i].name, search[i].value);
                }
                fprintf(fd,"\n************************************\n");
                est_criteria(search, searchidx, criteria);
                i=0;
                while(criteria[i].name != NULL) {
                        fprintf(fd,"%s = %s, %s\n", criteria[i].name, criteria[i].from,
criteria[i].to);
                        /* Perform error checking */
                        if(criteria[i].to == NULL) criteria[i].to = "0.0";
                        if(atof(criteria[i].to) < atof(criteria[i].from)) {
                                /* TO value equal zero? */
                                if(!(criteria[i].to = "0.0")) {
```

```c
                              printf("<H3><BLINK><I>ERROR - TO VALUE LESS
THAN FROM VALUE</I></BLINK></H3>\n");
                              printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                      }

              }
/*               printf("Name = %s,From = %s,To = %s\n",criteria[i].name,
criteria[i].from, criteria[i].to);*/
              i++;
      }


      /*                      */
      /*                      */
      /* Search the header files     */
      /*                      */
      /*                      */
      sprintf(list_name,"%ilist",getpid());  /* Create the list output CPI file name */
      sprintf(cmd,"/tmp/%s",list_name);
      hdr_search(criteria,hdr_path,cpi_path,fd,cmd,boolean);
      /*                      */
      /*                      */
      /* Output the search results   */
      /*                      */
      /*                      */
      system("cat /tmp/query_search >> /tmp/query_search_cum");
      system("cat /tmp/query_search | /usr/lib/sendmail -t
cavov@magnum.oc.rf.af.mil");
      fclose(fd); /* Close the query search file */
      /*
      Bug: using ! after > causes file to  be created but not written to. Just use >
      */
      /*        sprintf(cmd,"ls /xbox/disk0/cavovn/data | grep mat >
/tmp/%s",list_name);*/
      /*sprintf(cmd,"ls /xbox/disk0/cavovn/httpd/data | grep mat >
/tmp/%s",list_name);
      system(cmd);*/
/*        printf("<H3><BLINK><I>MATCHING MATLAB
FILES</I></BLINK></H3>\n"); */
      printf("<H3><I>MATCHING MATLAB FILES</I></H3>\n");
      printf("<P><STRONG>Please note that files ending with suffix 'gz' were \
      compressed using the utility 'gzip'. You will need 'gunzip' to uncompress \
      them.</STRONG><BR><BR>\n");
/*        printf("<FORM METHOD=\"POST\">\n");
      printf("<SELECT NAME=\"MCARM_Query_Results\" SIZE=10>\n");
      printf("<OPTION SELECTED>None\n"); */
      sprintf(cmd,"/tmp/%s",list_name);
```

```c
                if((fd_cpi = fopen(cmd,"r")) == '\0') {
                        if(no_rec_found) {  /* Matching records or files found? */
                                printf("main: Cannot open '%s' file\n", list_name);
                                exit();
                        }
                }
                else {
                        counter = 0;
                        printf("<PRE>");
                        while((fgets(line,PATHLEN,fd_cpi)) != NULL) {
                                for(i=0;line[i] != NULL;i++)  /* Get rid of newline character */
                                        if(line[i] == '\n')
                                                line[i] = '\0'; /* Replace newline with NULL
character */
                                counter = counter + 1;
                                strcpy(buff,(strrchr(line,'/')+1)); /* Strip off path information */
                                if(counter%MODULO_OP == 0)
                                {
                                        printf("<A
HREF=\"/~cavovn/cpi/%s\">%s</A>\n",buff, buff);
                                }
                                else
                                {
                                        printf("<A HREF=\"/~cavovn/cpi/%s\">%s</A>
",buff, buff);
                                }
/*                              printf("<OPTION><A
HREF=\"/~cavovn/cpi/%s\">%s</A>",buff, buff); */
/*                              printf("<OPTION><A
HREF=\"/~cavovn/cpi/%s\">%s</A><BR>",buff, buff); */
                        }
                        printf("</PRE>");
/*                      printf("</SELECT>\n</FORM>\n"); */
                }
                if(!no_rec_found) {  /* No matching records or files found? */
                        printf("<B><BLINK><I>NO MATCH</I></BLINK></B>\n");
                }
                else {
                        printf("<B><I>%d Data Files Found</I></B>\n",no_rec_found);
                }
/*              printf("<H3><BLINK><I>Thank you. Please select compressed Matlab file
to download.</I></BLINK></H3>\n"); */
                printf("<H3><I>Thank you. Please select compressed Matlab file to
download.</I></H3>\n");
/*              printf("<H3><BLINK><I>Select <EM>Back</EM> pointer to return to
query.</I></BLINK></H3>\n"); */
```

```c
                printf("<H3><I>Select <EM>Back</EM> pointer to return to
query.</I></H3>\n");
                fclose(fd_cpi);
        }
        else {
                printf("<H3><BLINK><I>NOTHING TO
SEARCH</I></BLINK></H3>\n");
        }
        printf("<a
href=\"http://sunrise.oc.rl.af.mil:80/~cavovn/forms/query_new.html\"><img align=\"left\"
src=\"/icons/point_11.gif\" alt=\"Back\"></a>\n");
        printf("</BODY></HTML>\n");
        system("/xbox/disk0/cavovn/bin/netscape
http://sunrise.oc.rl.af.mil/flight_path5.html");
/*
        Cleanup
*/
/*      sprintf(cmd,"rm -f /tmp/%s",list_name);*/
        system(cmd);    /* Remove the created list file */
}


split(buf,begin,end)
char buf[STRLEN]; /* stdin buffer */
int begin; /* beginning index pointer for name/value pair */
int end; /* end index pointer for name/value pair */
{
        register int i;

        for(i=begin;i<end;i++) {
                if(buf[i] == '=') {
                        buf[i] = '\0'; /* set equal to null to terminate name */
                        pairbuf[pairidx].name = &buf[begin];
                        pairbuf[pairidx].value = &buf[i+1];
                        return;
                }
        }
}

pack(buf,length)
char buf[STRLEN];
int length; /* string length in bytes */
{
        char temp[STRLEN];
        int i,j=0;

        for(i=0;i<length && buf[i] != '\0';i++) {
```

```
                    if(buf[i] == '?') continue; /* Ignore hexdecimal symbol */
                    else if(buf[i] != ' ')
                            temp[j++] = buf[i];
                    else {
                            if(buf[i+1] == ' ')
                                    continue;
                            else
                                    temp[j++] = buf[i];
                    }
            }
            temp[j] = '\0'; /* Null terminate temporary buffer */
            strcpy(buf,temp); /* Copy packed string into processing buffer */
    }


    est_criteria(inbuf, count, outbuf)
    struct pairs inbuf[MAX_PAIRS];
    int count;
    struct triples outbuf[MAX_PAIRS];
    {
            int i, j;

            j = 0;
            for(i=0;i<count;i=i+2) {
                    outbuf[j].name = inbuf[i].value; /* Get the parameter name */
                    outbuf[j].from = inbuf[i+1].value; /* Get the from value */
                    outbuf[j++].to = inbuf[i+2].value; /* Get the to value */
            }
    }


    hdr_search(criteria,hdr_path,cpi_path,fd,cpi_file,boolean)
    struct triples criteria[];
    char hdr_path[],cpi_path[];
    FILE *fd;
    char cpi_file[MAX_PAIRS];
    unsigned short boolean;
    {
    FILE *fd_hdr;
    char hdr_list[PATHLEN], cmd[MAX_PAIRS], line[PATHLEN], buff[PATHLEN],
    file_name[TEXT_SIZE];
    int i, len;

            sprintf(hdr_list,"%ihdr_list",getpid());  /* Construct the header directory list output
    file name */
            /* Imbedded newline in HDR_PATH caused must grief and  havoc. Causes system
    'cmd' not to work! */
            sprintf(cmd,"ls %s | grep \"\\.hdr\" > /tmp/%s",hdr_path, hdr_list);
```

```c
        system(cmd);
        sprintf(cmd,"/tmp/%s",hdr_list);
        fprintf(fd,"HDR_SEARCH %s\n",hdr_path);
        fprintf(fd,"HDR_SEARCH %s\n",cmd);
        if((fd_hdr = fopen(cmd,"r")) == '\0') {
                printf("hdr_search: Cannot open '%s' file\n", hdr_list);
                exit();
        }
        while((fgets(line,PATHLEN,fd_hdr)) != NULL) {
                fprintf(fd,"HDR_SEARCH %s\n",line);
                for(i=0;line[i] != NULL;i++)  /* Get rid of newline character */
                        if(line[i] == '\n')
                                line[i] = '\0'; /* Replace newline with NULL character */
                sprintf(file_name,"%s/%s",hdr_path,line); /* Construct header absolute file
name */

                /* read_hdr routine returns 0 if no match, 1 if match */
                if(read_hdr(criteria,file_name,boolean)) { /* Search header file for criteria
match */
                        strncpy(buff,line,(strcspn(line,"mat")-1));
/*                      sprintf(cmd,"ls %s/%s*CPI* >> %s",cpi_path, buff, cpi_file); */
                        sprintf(cmd,"ls %s/%s*.gz >> %s",cpi_path, buff, cpi_file);
                        ++no_rec_found; /* Increment number records found counter */
                        system(cmd);
                }
                len = strlen(buff);
                for(i=0;i<len;i++) buff[i] = '\0'; /* Clear the buffer */
        }
        fclose(fd_hdr); /* Close the header list file */
        sprintf(cmd,"rm /tmp/%s",hdr_list);
        system(cmd);    /* Remove the created header list file */
}
```

Appendix C – Software Tools

Software tools were acquired or written to facilitate the construction, operation and maintenance of the MCARM Web pages. In all instances, only public domain software was identified and acquired to contain cost and to insure portability. In some cases, C programs had to be written to fulfill special requirements. This appendix documents these software tools and provides source listings for the developed code.

Tool	Description
account_request	Request a new MCARM account
accesswatch	Perl program to display daily or to date access statistics to track site usage
column	C program to replicate text columns
comments_form	Process user MCARM comments
gcc	C/C++ compiler
gifmerge	Used to merge GIF 89a files for animation creation
giftrans	Translates GIF 87 to GIF 89a files for animation creation
grass	Geographic Information System (GIS) mapping software
gunzip, gzip	Decompression/Compression software
hotmetal	HTML editor
mapedit	Image map creation/editor
matlab_hdr	C program to read and extract header information from the MCARM Matlab files. Creates distinct header files needed for searching the data base.
mpeg_play	MPEG player
mpeginfo	MPEG information
netscape	Internet browser
perl	Programming langauge interpreter
query_form	Query the MCARM data base
xanim	X-Windows animation display tool
xemacs	X-Windows visual editor
xpaint	X-Windows paint program used to create image maps, logos and animation
xplay	X-Windows audio player
xv	X-Windows image viewer used for image viewing, translation, and manipulation

Table C-1 Software Tools

```
/*
#
#
#
#    MCARM Account Request Form
#
#
# created 4/30/96 author: V.N. Cavo
#
*/
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define STRLEN 10000   /* Define stdin buffer size parameter */
#define MAX_PAIRS 10000   /* Define number of name/value pairs parameter */

struct pairs {
        char *name;
        char *value;
} pairbuf[MAX_PAIRS];   /* allow for MAX_PAIRS name/value pairs */

int pairidx=0;
FILE *fopen(), *fd;

main(argc, argv)
int argc;
char *argv[];
{
        char buffer[STRLEN], *ptr, *ctime(), datebuf[26], line[80], *fgets();
        register int len, clen, i, j, startptr;
        long clock, time();

        /* get system date and time for timestamp */
        clock=0;
        time(&clock);
        ptr = ctime(&clock);
        strncpy(datebuf,ptr,26);
        printf("%s\n",datebuf);
        /*
        */
        /* process POST(recommended) method only */
        if(strcmp(getenv("REQUEST_METHOD"),"POST")) exit();
        /* ignore url encode forms */
        if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
exit();
        /* get stdin length from environment variable */
```

```c
#ifdef OLD /* support for non-CGI server (e.g. NCSA 1.0a5 or earlier) which used
            command-line arguments */
        clen = atoi(argv[1]);
#else   /* get string from environment variable */
        clen = atoi(getenv("CONTENT_LENGTH"));
#endif
        /* print the CGI required header information */
        /* specify full document and MIME type - Note: blank line required */
        /* printf("Content-type: text/html\n\n");*/
        /* print document title and initial heading */
        printf("<HTML>\n");
        printf("<HEAD><TITLE>MCARM Account Request Form
Response</TITLE></HEAD>\n");
        printf("<BODY><H1>Account Request Response</H1>\n");
        gets(buffer);  /* get the stdin input user response buffer */
        len = strlen(buffer);
/*      if(len != clen) printf("len=%d, clen=%d\n",len,clen); */
/*      printf("%s\nlen=%d\n",buffer,len);  */
        buffer[len] = '&';  /* Terminate last name/value pair */
        buffer[++len] = '\0'; /* Terminate buffer string */
/*      printf("%s\nlen=%d\n",buffer,len);  */
        /* Replace '+' and hexadecimal special characters */
        for(i=0;i<len;i++) {
                if(buffer[i] == '+') buffer[i] = ' ';
                if(buffer[i] == '%') { /* hex special character? */
                        buffer[i] = '?';  /* prepare for packing */
                        buffer[i+1] = '?';
                        buffer[i+2] = '?';
                }
        }
        /* Pack the buffer to remove redundant blanks */
        pack(buffer, len);
        /*                              */
        /* Split the buffer into name/value pairs */
        /*                              */
        len = strlen(buffer);  /* Get new string length */
        startptr = 0;
        for(i=0;i<len;i++) {
                if(buffer[i] == '&') {
                        buffer[i] = '\0'; /* set ampersand to null */
                        split(buffer,startptr,i);
                        startptr = i+1;
                        if((++pairidx) > MAX_PAIRS) break;
                }
        }
        /*                      */
```

```c
        /*                      */
        /* Process the form information */
        /*                      */
        /*                      */
        if((fd = fopen("/tmp/account_request","w")) == '\0') {
                fprintf(fd,"Cannot open temporary file '/tmp/account_request");
                exit();
        }
        /* Print timestamp and header information */
        fprintf(fd,"%s\n\n", datebuf);
        /* Print user supplied information */
        for(i=0;i<pairidx;i++) {
                /* Push the parameter name onto data base stack */
/*      retquote(pairbuf[i].name); ++nargs; */
                for(j=0;pairbuf[i].value[j] != '\0';j++)
                    if(islower(pairbuf[i].value[j]))
                            pairbuf[i].value[j] = toupper(pairbuf[i].value[j]);
                /* Push the parameter value onto data base stack */
/*              retquote(pairbuf[i].value); ++nargs; */
                fprintf(fd,"%s = %s\n", pairbuf[i].name, pairbuf[i].value);
        }
        fprintf(fd,"\n*************************************\n");
        fclose(fd); /* Close the temporary file */
        system("cat /tmp/account_request >> /tmp/account_request_cum");
        system("cat /tmp/account_request | /usr/lib/sendmail -t cavovn@sunrise.oc.rl.af.mil");
        printf("<H2><BLINK>Thank you for your MCARM account request. We will
contact you within a few days at the supplied telephone number.</BLINK></H2/>\n");
        close(fd); /* Close output file */
        printf("</BODY></HTML>\n");
}

split(buf,begin,end)
char buf[STRLEN]; /* stdin buffer */
int begin; /* beginning index pointer for name/value pair */
int end; /* end index pointer for name/value pair */
{
        register int i;

        for(i=begin;i<end;i++) {
                if(buf[i] == '=') {
                        buf[i] = '\0'; /* set equal to null to terminate name */
                        pairbuf[pairidx].name = &buf[begin];
                        pairbuf[pairidx].value = &buf[i+1];
                        return;
                }
        }
```

```c
}

pack(buf,length)
char buf[STRLEN];
int length; /* string length in bytes */
{
        char temp[STRLEN];
        int i,j=0;

        for(i=0;i<length && buf[i] != '\0';i++) {
                if(buf[i] == '?') continue; /* Ignore hexdecimal symbol */
                else if(buf[i] != ' ')
                        temp[j++] = buf[i];
                else {
                        if(buf[i+1] == ' ')
                                continue;
                        else
                                temp[j++] = buf[i];
                }
        }
        temp[j] = '\0'; /* Null terminate temporary buffer */
        strcpy(buf,temp); /* Copy packed string into processing buffer */
}
```

```
/*
#
#
#
#    Function to copy columns
#
#
# created 7/26/96 author: V.N. Cavo
#
# This function comes in handy when it is necessary to copy columns
# within a file (e.g. to create a script for renaming a large number
# of files).
#
*/

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <ctype.h>

#define TEXT_SIZE 500

main(argc, argv)
int argc;
char *argv[];
{
        int i;
        char *fgets(), text[TEXT_SIZE];
        FILE *fp;

        while (--argc > 0)  /* process all of the specified files */
        {
                if((fp = fopen(*++argv,"r")) == '\0') { /* Open file for reading */
                        printf("Cannot open '%s' file\n", *argv);
                        exit();
                }
                while((fgets(text,TEXT_SIZE,fp)) != NULL) {
                        for(i=0;text[i] != NULL;i++)  /* Get rid of newline character */
                                if(text[i] == '\n')
                                        text[i] = '\0'; /* Replace newline with NULL character
*/
                        printf("%s %s\n", text, text);
                }
        }
}
```

```
/*

#
#
#
#    MCARM Comments Form
#
#
# created 1/23/96 author: V.N. Cavo

# last modified on 2/18/97 - Modified e-mail address for sendmail
#
# cc comments_form.c -o comments_form
#
*/
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define STRLEN 10000    /* Define stdin buffer size parameter */
#define MAX_PAIRS 10000   /* Define number of name/value pairs parameter */

struct pairs {
        char *name;
        char *value;
} pairbuf[MAX_PAIRS];   /* allow for MAX_PAIRS name/value pairs */

int pairidx=0;
FILE *fopen(), *fd;

/*int query_prep(nargs)*/
/*int nargs;*/
main(argc, argv)
int argc;
char *argv[];
{
        char buffer[STRLEN], *ptr, *ctime(), datebuf[26], line[80], *fgets();
        register int len, clen, i, j, startptr;
        long clock, time();

        /* get system date and time for timestamp */
        clock=0;
        time(&clock);
        ptr = ctime(&clock);
        strncpy(datebuf,ptr,26);
        printf("%s\n",datebuf);
```

```c
/*
*/
/* process POST(recommended) method only */
if(strcmp(getenv("REQUEST_METHOD"),"POST")) exit();
/* ignore url encode forms */
if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
exit();
/* get stdin length from environment variable */
#ifdef OLD /* support for non-CGI server (e.g. NCSA 1.0a5 or earlier) which used
          command-line arguments */
    clen = atoi(argv[1]);
#else   /* get string from environment variable */
    clen = atoi(getenv("CONTENT_LENGTH"));
#endif
    /* print the CGI required header information */
    /* specify full document and MIME type - Note: blank line required */
    /* printf("Content-type: text/html\n\n");*/
    /* print document title and initial heading */
    printf("<HTML>\n");
    printf("<HEAD><TITLE>MCARM Comments Form
Response</TITLE></HEAD>\n");
    printf("<BODY><H1>Comments Response</H1>\n");
    gets(buffer);  /* get the stdin input user response buffer */
    len = strlen(buffer);
/*      if(len != clen) printf("len=%d, clen=%d\n",len,clen); */
/*      printf("%s\nlen=%d\n",buffer,len);  */
    buffer[len] = '&';  /* Terminate last name/value pair */
    buffer[++len] = '\0'; /* Terminate buffer string */
/*      printf("%s\nlen=%d\n",buffer,len);  */
    /* Replace '+' and hexadecimal special characters */
    for(i=0;i<len;i++) {
            if(buffer[i] == '+') buffer[i] = ' ';
            if(buffer[i] == '%') { /* hex special character? */
                    buffer[i] = '?';  /* prepare for packing */
                    buffer[i+1] = '?';
                    buffer[i+2] = '?';
            }
    }
    /* Pack the buffer to remove redundant blanks */
    pack(buffer, len);
    /*                      */
    /* Split the buffer into name/value pairs */
    /*                      */
    len = strlen(buffer);  /* Get new string length */
    startptr = 0;
    for(i=0;i<len;i++) {
```

```c
            if(buffer[i] == '&') {
                    buffer[i] = '\0'; /* set ampersand to null */
                    split(buffer,startptr,i);
                    startptr = i+1;
                    if((++pairidx) > MAX_PAIRS) break;
            }
    }
    /*                      */
    /*                      */
    /* Process the form information */
    /*                      */
    /*                      */
    if((fd = fopen("/tmp/comments_search","w")) == '\0') {
            fprintf(fd,"Cannot open temporary file '/tmp/comments_search");
            exit();
    }
    /* Print timestamp and header information */
    fprintf(fd,"%s\n\n", datebuf);
    /* Print user supplied information */
    for(i=0;i<pairidx;i++) {
            /* Push the parameter name onto data base stack */
/*      retquote(pairbuf[i].name); ++nargs; */
            for(j=0;pairbuf[i].value[j] != '\0';j++)
                if(islower(pairbuf[i].value[j]))
                        pairbuf[i].value[j] = toupper(pairbuf[i].value[j]);
            /* Push the parameter value onto data base stack */
/*              retquote(pairbuf[i].value); ++nargs; */
            fprintf(fd,"%s = %s\n", pairbuf[i].name, pairbuf[i].value);
    }
    fprintf(fd,"\n*************************************\n");
    fclose(fd); /* Close the temporary file */
    system("cat /tmp/comments_search >> /tmp/comments_search_cum");
    system("cat /tmp/comments_search | /usr/lib/sendmail -t
cavovn@magnum.oc.rl.af.mil");
    printf("<H2><BLINK>Thank you for your comments/questions. Be assured that they
will receive prompt attention.</BLINK></H2/>\n");
    close(fd); /* Close output file */
    printf("</BODY></HTML>\n");
/*      retint(nargs/2); /* Push the number of name/value pairs onto stack */
/*      return(++nargs); /* Return total number of stack elements */
}


split(buf,begin,end)
char buf[STRLEN]; /* stdin buffer */
int begin; /* beginning index pointer for name/value pair */
int end; /* end index pointer for name/value pair */
```

54

```
{
        register int i;

        for(i=begin;i<end;i++) {
                if(buf[i] == '=') {
                        buf[i] = '\0'; /* set equal to null to terminate name */
                        pairbuf[pairidx].name = &buf[begin];
                        pairbuf[pairidx].value = &buf[i+1];
                        return;
                }
        }
}


pack(buf,length)
char buf[STRLEN];
int length; /* string length in bytes */
{
        char temp[STRLEN];
        int i,j=0;

        for(i=0;i<length && buf[i] != '\0';i++) {
                if(buf[i] == '?') continue; /* Ignore hexdecimal symbol */
                else if(buf[i] != ' ')
                        temp[j++] = buf[i];
                else {
                        if(buf[i+1] == ' ')
                                continue;
                        else
                                temp[j++] = buf[i];
                }
        }
        temp[j] = '\0'; /* Null terminate temporary buffer */
        strcpy(buf,temp); /* Copy packed string into processing buffer */
}
```

```c
/*
#
#
#
#    MCARM MATLAB Header Routine
#
#

# created 2/6/96 author: V.N. Cavo
# modified 2/8/96 by V.N. Cavo

#
# modified 6/18/96 - Changed code to generate only the header file.
#
# Compilation: cc matlab_hdr.c -o matlab_hdr
#
*/
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <fcntl.h>
#include <ctype.h>
/*#include <stdlib.h>*/


#define BUF_SIZE 9
#define TEXT_SIZE 500
#define PERMS 0777
#define REOF 0

typedef struct {
        long type; /* type of form MOPT */
        long mrows; /* row dimension */
        long ncols; /* column dimension */
        long imagf; /* flag indicating imaginary part */
        long namlen;/* namlen length (including NULL - i.e. 1+ the name length)*/
} Fmatrix;   /* allow for MAX_PAIRS name/value pairs */

char *pname;     /* pointer to matrix name */
double *pr;      /* pointer to real data */
double *pi;      /* pointer to imaginary data */
FILE *fp;
Fmatrix x;
int mn;
static int element_size[] = { 8, 4, 4, 2, 2, 1}; /* Element size in bytes: 64-bit double float, 32-
bit float, \
```

32-bit signed integer, 16-bit signed integer,

16-bit unsigned \

integer, 8-bit unsigned integer */

```c
main(argc, argv)
int argc;
char *argv[];
{
        char *ptr, *ctime(), datebuf[26], *fgets(), text[TEXT_SIZE],
hdr_filename[TEXT_SIZE], cpi_filename[TEXT_SIZE], cmd[TEXT_SIZE];
        register int len, clen, i, j, startptr;
        long clock, time();
        double real_data[BUF_SIZE], imag_data[BUF_SIZE];
        int nitems, done=0, no_bytes, m, o, p, t;
        long float fvalue, *ptf;
        int ivalue, *pti, fd_hdr, fd_cpi;
        double text_data[TEXT_SIZE], dvalue, *ptd;
        short svalue, *pts;
        unsigned short usvalue, *ptus;
        unsigned char cvalue, *ptc;
        unsigned int c;
        unsigned long word;
        FILE *fp_hdr, *fp_cpi, *fpw, *fp_temp;

        /* get system date and time for timestamp */
        clock=0;
        time(&clock);
        ptr = ctime(&clock);
        strncpy(datebuf,ptr,26);
        printf("%s\n",datebuf);
        /*
        */

        while (--argc > 0)  /* process all of the specified files */
        {
          done = 0;
          if(strstr(*++argv, ".mat") == NULL) { /* MATLAB file? */
                printf("File '%s' not a MATLAB file -- skipped\n\n", *argv);
                continue;
          }
          else if((strstr(*argv, ".gz") != NULL) || (strstr(*argv, ".Z") != NULL)) /*
Compressed MATLAB file? */
                {
                        printf("File '%s' is a compressed MATLAB file --\n", *argv);
                        printf("Please decompress and try again (skipped)\n\n");
                        continue;
```

```c
                      }
          else
          {
                  if((fp = fopen(*argv,"r")) == '\0') { /* Open file for reading */
                        printf("Cannot open '%s' file\n", *argv);
                        exit();
                  }
                  sprintf(hdr_filename,"%s.hdr",*argv); /* Construct the header file name */
                  /* Create the header file, if it doesn't exist; otherwise an error */
                  if((fd_hdr = open(hdr_filename, O_RDWR | O_CREAT | O_EXCL,
PERMS)) == -1) {
                        printf("Cannot create '%s' header file\n", hdr_filename);
                        exit();
                  }
                  if((fp_hdr = fdopen(fd_hdr,"w")) == '\0') { /* Open file stream for writing
*/
                        printf("Cannot open '%s' file stream\n", hdr_filename);
                        exit();
                  }
                  while(!done) /* Do forever */
                  {
                        nitems = fread(&x, sizeof(Fmatrix), 1, fp);  /*Read the header record*/
                        if(nitems == REOF) {
                              done = 1;
                              break; /*End of file?*/
                        }
/*                      printf("Type = %d Namlen = %d\n", x.type, x.namlen);*/
                        pname = &datebuf[0];
                        nitems = fread(pname, sizeof(char), x.namlen, fp);  /* Read the
variable name */
                        if(nitems == REOF) {
                              done = 1;
                              break; /*End of file?*/
                        }
                        get_type(x.type, &no_bytes, &p, &t);  /* Determine no. bytes to read
& matrix type */
/*                      printf("%s = ", pname);*/
                        if((strncasecmp(pname, "CPI",3) != 0) || !isdigit((unsigned
int)pname[3])) { /* CPI data? */
                              fpw = fp_hdr; /* Set file pointer to write header data */
                        }
                        else {  /* CPI data found */
                              sprintf(cpi_filename,"%s.%s",*argv,pname);  /* Construct the
CPI file name */

                              /* Create the CPI file, if it doesn't exist; otherwise an error */
```

58

```c
                    if((fd_cpi = open(cpi_filename, O_RDWR | O_CREAT |
O_EXCL, PERMS)) == -1) {
                            printf("Cannot create '%s' CPI file\n", cpi_filename);
                            exit();
                    }
                    if((fp_cpi = fdopen(fd_cpi,"w")) == '\0') { /* Open file stream
for writing */
                            printf("Cannot open '%s' file stream\n", cpi_filename);
                            exit();
                    }
                    fpw = fp_cpi; /* Set file pointer to write CPI data */
            }
            mn = x.mrows * x.ncols;
            nitems = fwrite(&x, sizeof(Fmatrix), 1, fpw);  /*Write the header
record*/

            nitems = fwrite(pname, sizeof(char), x.namlen, fpw);  /* Write the
variable name */

            if(nitems == REOF) {
                    done = 1;
                    break; /*End of file?*/
            }
/*          printf("mn=%i\n",mn);*/
/*          printf("P=%i, T=%i\n",p,t);*/
            if (!t) {  /* Numeric information */
                    while(mn-- > 0)
                    {
                            switch(p) {
                                    case 0: /* Print double precision (64-bit)
floating point */
                                            ptd = &dvalue;
                                            nitems = fread(ptd, no_bytes, 1, fp);  /*
Read the variable name */
                                            if(nitems == REOF) {
                                                    done = 1;
                                                    break; /*End of file?*/
                                            }
                                            nitems = fwrite(ptd, no_bytes, 1, fpw);
/* Write the variable name */
/*                                          printf("%e\n", *ptd);*/
                                            break;
                                    case 1: /* Print single precision (32-bit) floating
point */
                                            /*
                                            Note: variable 'fvalue' must be
defined as type 'long' so that
```

59

create a float type. Defining

will NOT work. The only other

instead of function 'fread' as

int and word as unsigned long.

Read the variable name */

/* Write the variable name */
/*

Read the variable name */

/* Write the variable name */
/*

signed) */

Read the variable name */

bytes are properly loaded to

'fvalue' (aka 'ptf') as just float

choice is to use function 'getc'

follows:

```
for(i=0;i<no_bytes;i++) {
        c = getc(fp);
        word = word << 8;
        word = word | c;
}
```
where c is defined as unsigned

```
*/
ptf = &fvalue;
nitems = fread(ptf, no_bytes, 1, fp);  /*

if(nitems == REOF) {
        done = 1;
        break; /*End of file?*/
}
nitems = fwrite(ptf, no_bytes, 1, fpw);

printf("%e\n", *ptf);*/
break;
case 2:  /* Print 32-bit signed integer */
        pti = &ivalue;
        nitems = fread(pti, no_bytes, 1, fp);  /*

if(nitems == REOF) {
        done = 1;
        break; /*End of file?*/
}
nitems = fwrite(pti, no_bytes, 1, fpw);

printf("%i\n", *pti);*/
break;
case 3:  /* Print short (16-bit) signed integer

pts = &svalue;
nitems = fread(pts, no_bytes, 1, fp);  /*

if(nitems == REOF) {
```

```
                                done = 1;
                                break; /*End of file?*/
                        }
                        nitems = fwrite(pts, no_bytes, 1, fpw);

                        printf("%i\n", *pts);*/
                        break;
                case 4: /* Print unsigned short (16-bit) integer
```

/* Write the variable name */
/*

*/

```
                        ptus = &usvalue;
                        nitems = fread(ptus, no_bytes, 1, fp);  /*
```

Read the variable name */

```
                        if(nitems == REOF) {
                                done = 1;
                                break; /*End of file?*/
                        }
                        nitems = fwrite(ptus, no_bytes, 1, fpw);

                        printf("%u\n", *ptus);*/
                        break;
                case 5: /* Print unsigned character (8-bit
```

/* Write the variable name */
/*

unsigned integer) */

Read the variable name */

```
                        ptc = &cvalue;
                        nitems = fread(ptc, no_bytes, 1, fp);  /*
```

```
                        if(nitems == REOF) {
                                done = 1;
                                break; /*End of file?*/
                        }
                        nitems = fwrite(ptc, no_bytes, 1, fpw);
```

/* Write the variable name */
/*

```
                        printf("%u\n", *ptc);*/
                        break;
                }  /* End of real data switch */
                if(x.imagf) {
                        switch(p) {
                        case 0:  /* Print double precision (64-bit)
```

floating point */

```
                                ptd = &dvalue;
                                nitems = fread(ptd, no_bytes, 1,
```

fp); /* Read the variable name */

```
                                if(nitems == REOF) {
                                        done = 1;
                                        break; /*End of file?*/
                                }
```

fpw); /* Write the variable name */
/*

floating point */

must be defined as type 'long' so that

to create a float type. Defining

NOT work. The only other choice is to

of function 'fread' as follows:

8;

unsigned int and word as unsigned long.

fp); /* Read the variable name */

fpw); /* Write the variable name */
/*

fp); /* Read the variable name */

```
           nitems = fwrite(ptd, no_bytes, 1,

           printf("%e\n", *ptd);*/
           break;
case 1:  /* Print single precision (32-bit)

     /*
              Note: variable 'fvalue'

              bytes are properly loaded

              'fvalue' (aka 'ptf') will

              use function 'getc' instead


              for(i=0;i<no_bytes;i++) {
                     c = getc(fp);
                     word = word <<

                     word = word | c;
              }
               where c is defined as

      */
      ptf = &fvalue;
      nitems = fread(ptf, no_bytes, 1,

      if(nitems == REOF) {
              done = 1;
              break; /*End of file?*/
      }
      nitems = fwrite(ptf, no_bytes, 1,

      printf("%e\n", *ptf);*/
      break;
case 2:  /* Print 32-bit signed integer */
      pti = &ivalue;
      nitems = fread(pti, no_bytes, 1,

      if(nitems == REOF) {
              done = 1;
              break; /*End of file?*/
      }
```

```
fpw);  /* Write the variable name */
/*



integer signed) */



fp);  /* Read the variable name */






fpw);  /* Write the variable name */
/*



integer */



fp);  /* Read the variable name */






fpw);  /* Write the variable name */
/*



unsigned integer) */



fp);  /* Read the variable name */






fpw);  /* Write the variable name */
/*



                                nitems = fwrite(pti, no_bytes, 1,

                                printf("%i\n", *pti);*/
                                break;
        case 3:  /* Print short (16-bit) signed

                                pts = &svalue;
                                nitems = fread(pts, no_bytes, 1,

                                if(nitems == REOF) {
                                        done = 1;
                                        break; /*End of file?*/
                                }
                                nitems = fwrite(pts, no_bytes, 1,

                                printf("%i\n", *pts);*/
                                break;
        case 4:  /* Print unsigned short (16-bit)

                                ptus = &usvalue;
                                nitems = fread(ptus, no_bytes, 1,

                                if(nitems == REOF) {
                                        done = 1;
                                        break; /*End of file?*/
                                }
                                nitems = fwrite(ptus, no_bytes, 1,

                                printf("%u\n", *ptus);*/
                                break;
        case 5:  /* Print unsigned character (8-bit

                                ptc = &cvalue;
                                nitems = fread(ptc, no_bytes, 1,

                                if(nitems == REOF) {
                                        done = 1;
                                        break; /*End of file?*/
                                }
                                nitems = fwrite(ptc, no_bytes, 1,

                                printf("%u\n", *ptc);*/
                                break;
                } /* End of switch */
            } /* End imaginary if */
        } /* End while */
```

```c
                    }
                    else {  /* Text information */
                            ptd = &text_data[0];
                            nitems = fread(ptd, no_bytes, mn, fp);  /* Read the variable
name */
                            if(nitems == REOF) {
                                    done = 1;
                                    break; /*End of file?*/
                            }
                            nitems = fwrite(ptd, no_bytes, mn, fpw);  /* Write the variable
name */
                            for(i=0;i<nitems;i++) {
                                    text[i] = (char)(text_data[i]);  /* Force conversion to
character */
                                    if(text[i] == NULL) text[i] = ' ';  /* Eliminate redundant
NULLs */
                            }
                            text[nitems] = '\n';  /* Terminate string */
/*                          printf("%s\n", text);*/
                            for(i=0;i<nitems;i++) text[i] = 0;  /* Clear the array */
                    }  /* End of text information else */
            }  /* Not done while */
        }  /* End of else */
        fclose(fp); /* Close the source read file */
        fclose(fp_hdr); /* Close the associated header file */
        fclose(fp_cpi); /* Close the associated CPI file */
        sprintf(cmd,"rm -f %s", cpi_filename); /* Quick fix - delete the CPI file*/
        system(cmd);
    }  /* End of outer while statement */
    return;
}

get_type(type, ptr_bytes, p, t)
long type;
int *ptr_bytes, *p, *t;
{
    int i, value, m, o;

    i = 0;
    value = type;
    while(i<4) {
            switch(i){
                    case 0:
                            *t = value%10;
                            value = value/10;
                            break;
```

64

```
                case 1:
                        *p = value%10;
                        value = value/10;
                        *ptr_bytes = element_size[*p];
                        break;
                case 2:
                        o = value%10;
                        value = value/10;
                        break;
                case 3:
                        m = value%10;
                        value = value/10;
                        break;
        }
        ++i;

    }
}
```

```
/*
#
#
#
#    MCARM Inquiry Form
#
#
# created 12/5/95 author: V.N. Cavo
# last modified on 7/24/95
#
# 6/18/96 - Modified software to display the Matlab compressed file (.gz) instead of the just
#       the CPI suffixed file. The compressed file contains the header information. The
#       environment variable for CPI_PATH was not changed. It now points to the location
#       of the compressed Matlab files.
#
# 7/24/96 - Modified software to incorporate value range capability. Also, corrected a bug
#       where an erroneous error was received when no match was found.
#
# 11/20/96 - Took message blink out and updated below message.
#
# Note: read_hdr.c routine must be compiled with query_form.c
#     cc query_form_new.c read_hdr_new.c -o query_form_new
#     The query_form executable must then be placed in 'cgi-bin' where
#     it is invoked by 'query_new.html' upon query submission.
*/
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <ctype.h>

#define PERMS 0666
#define STRLEN 5000      /* Define stdin buffer size parameter */
#define MAX_PAIRS 1000   /* Define number of name/value pairs parameter */
#define PATHLEN 100      /* Define environment variable HDR_PATH length */
#define TEXT_SIZE 500    /* Concatenated file name string */
#define MODULO_OP 4      /* Number of files per line to display */

struct pairs {
        char *name;            /* Parameter name */
        char *value;           /* Parameter value */
        short int condition;   /* Criteria True(=1) or False(=0) condition */
} pairbuf[MAX_PAIRS], search[MAX_PAIRS];  /* allow for MAX_PAIRS name/value
pairs */

struct triples {
        char *name;            /* Parameter name */
```

```c
        char *from;                /* Parameter from value */
        char *to;                  /* Parameter to value */
        short int condition;       /* Criteria True(=1) or False(=0) condition */
} criteria[MAX_PAIRS];             /* allow for MAX_PAIRS name/from/to triplets */

int pairidx=0, searchidx=-1, getpid(), no_rec_found=0, none_flag=0;
char cmd[MAX_PAIRS], list_name[PATHLEN]; /* System call command string */
FILE *fopen(), *fd, *fd_env, *fd_cpi;
extern short int read_hdr();

main(argc, argv)
int argc;
char *argv[];
{

char
buffer[STRLEN],*ptr,*ctime(),datebuf[26],line[PATHLEN],buff[PATHLEN],*fgets(),hdr_p
ath[PATHLEN],cpi_path[PATHLEN];
register int len, clen, i, j, startptr, counter;
long clock, time();
unsigned short boolean=1; /* Set logical operator 1 (AND) 0 (OR) */

        /* get system date and time for timestamp */
        clock=0;
        time(&clock);
        ptr = ctime(&clock);
        strncpy(datebuf,ptr,26);
        printf("%s\n",datebuf);

        /* Open the output file */
        if((fd = fopen("/tmp/query_search","w")) == '\0') {
                fprintf(fd,"main: Cannot open temporary file '/tmp/query_search");
                exit();
        }
        /* Open the environment file and get HDR_PATH and CPI_PATH */
        /* Could not set HDR_PATH and CPI_PATH in CGI environment. Hence, next best
thing was
        to read these environment variables from a file to preclude recompilation.
        Note: CGI environment is different then user environment (i.e. env > /tmp/env_path
        puts CGI env into target file. That is, CGI sends its own environment to server. */
        if((fd_env = fopen("/xbox/disk0/cavovn/httpd/htdocs/env_path","r")) == '\0') {
                fprintf(fd,"main: Cannot open temporary file
'/xbox/disk0/cavovn/httpd/htdocs/env_path");
                exit();
        }
        /* Set the Header and CPI path environment variables */
```

```c
        fgets(line,PATHLEN,fd_env);
        strcpy(hdr_path,(strpbrk(line,"=")+1));
        for(i=0;hdr_path[i] != NULL;i++)  /* Get rid of newline character */
                if(hdr_path[i] == '\n')
                        hdr_path[i] = '\0'; /* Replace newline with NULL character */
        fprintf(fd,"HDR_PATH = %s\n",hdr_path);
        fgets(line,PATHLEN,fd_env);
        strcpy(cpi_path,(strpbrk(line,"=")+1));
        for(i=0;cpi_path[i] != NULL;i++)  /* Get rid of newline character */
                if(cpi_path[i] == '\n')
                        cpi_path[i] = '\0'; /* Replace newline with NULL character */
        fprintf(fd,"CPI_PATH = %s\n",cpi_path);
        fclose(fd_env); /* Close the environment file */
        /* Create the header and CPI directory links */
/*      sprintf(cmd,"unalias rm;cd /xbox/disk0/cavovn/public_html;rm -f cpi hdr",hdr_path);
/* delete Header/cpi path link */
        sprintf(cmd,"unalias rm;cd /xbox/disk0/cavovn/public_html;rm -f cpi hdr"); /* delete
Header/cpi path link */
        system(cmd);
        sprintf(cmd,"cd /xbox/disk0/cavovn/public_html;ln -s %s hdr",hdr_path); /* Header
path link */
        system(cmd);
        sprintf(cmd,"cd /xbox/disk0/cavovn/public_html;ln -s %s cpi",cpi_path); /* CPI path
link */
        system(cmd);


        /*
                Process environment variables
        */
        /* process POST(recommended) method only */
        if(strcmp(getenv("REQUEST_METHOD"),"POST")) exit();
        /* ignore url encode forms */
        /* Output of a form being processed? */
        if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
exit();
/*      Some variables and constructs are described as being `URL-encoded'.  In a URL
        encoded string an escape sequence consists of a percent character ("%")
        followed by two hexadecimal digits, where the two hexadecimal digits form
        an octet. An escape sequence represents the graphic character which has the
        octet as its code within the US-ASCII coded character set, if it exists. If
        no such graphic character exists, then the escape sequence represents the
        octet value itself.
*/
/*      strcpy(line,getenv("HTTP_ACCEPT"));
        printf("HTTP_ACCEPT = %s\n",line);*/
```

```c
        /* get stdin length from environment variable */
#ifdef OLD /* support for non-CGI server (e.g. NCSA 1.0a5 or earlier) which used
            command-line arguments */
        clen = atoi(argv[1]);
#else   /* get string from environment variable */
        clen = atoi(getenv("CONTENT_LENGTH"));
#endif

        if(clen > STRLEN) {
                printf("Buffer overrun\n");
                exit();
        }
        /*
                Output the HTML information
        */
        /* print the CGI required header information */
        /* specify full document and MIME type - Note: blank line required */
/*      printf("Content-type: text/html\n\n");*/
/*      printf("Content-type: application/x-gzip\n\n");*/
        /* print document title and initial heading */
        printf("<HTML>\n");
        printf("<HEAD><TITLE>MCARM Inquiry Form Response</TITLE></HEAD>\n");
        printf("<BODY><H1><IMG
SRC=\"/icons/mcarm_small.gif\">MCARM<BR>Inquiry Response</H1>\n");
        gets(buffer);  /* get the stdin input user response buffer - from HTML Form */
        /* Bug - strlen function reports 1 more character than reported by environment
variable
                CONTENT_LENGTH. Unexpected results occur if value returned by strlen
used!
                Work around is to use value reported by CONTENT_LENGTH. The function
strlen may be
                counting the terminating NULL character, contrary to documentation. */
        len = strlen(buffer);
/*      if(len != clen) printf("len=%d, clen=%d\n",len,clen); */
/*      printf("%s\nlen=%d\n",buffer,len);  */
        buffer[clen] = '&';  /* Terminate last name/value pair */
        buffer[len] = '\0'; /* Terminate buffer string */
        /* Replace '+' and hexadecimal special characters */
        for(i=0;i<len;i++) {
                if(buffer[i] == '+') buffer[i] = ' ';
                if(buffer[i] == '%') { /* hex special character? */
                        buffer[i] = '?';  /* prepare for packing */
                        buffer[i+1] = '?';
                        buffer[i+2] = '?';
                }
```

69

```
        }
        /* Pack the buffer to remove redundant blanks */
        pack(buffer, len);
        /*                              */
        /* Split the buffer into name/value pairs */
        /*                              */
        len = strlen(buffer);  /* Get new string length */
        startptr = 0;
        for(i=0;i<len;i++) {
                if(buffer[i] == '&') {
                        buffer[i] = '\0'; /* set ampersand to null */
                        split(buffer,startptr,i);
                        startptr = i+1;
                        if((++pairidx) > MAX_PAIRS) break;
                }
        }
        /*                      */
        /*                      */
        /* Process the form information */
        /*                      */
        /*                      */
        /* Print timestamp and header information */
        fprintf(fd,"%s\n\n", datebuf);
        /* Process user supplied information and perform some error checking */
        for(i=0;i<pairidx;i++) {
                for(j=0;pairbuf[i].value[j] != '\0';j++) /* Convert all characters to uppercase */
                    if(islower(pairbuf[i].value[j]))
                            pairbuf[i].value[j] = toupper(pairbuf[i].value[j]);
                if(((i % 3) == 0)) /* Parameter NONE specified during parameter name/value
pair cycle? */
                        if((!strcmp(pairbuf[i].value, "NONE")))
                                none_flag = 1;
                        else
                                none_flag = 0;
                /* Push the parameter value onto data base stack - place holder for DBMS */
                if(!strcmp(pairbuf[i].value, "AND")) /* Set boolean to AND */
                        boolean = 1;
                else if(!strcmp(pairbuf[i].value, "OR")) /* Set boolean to OR */
                        boolean = 0;
                /* Save parameters if not 'NONE' or NULL */
                else if(strcmp(pairbuf[i].value, "NONE") && strcmp(pairbuf[i].value, "\0")) {
                        if(none_flag) {
                        /* Some parameter 'NONE' set to a value */
                                printf("<H3><BLINK><I>ERROR - PARAMETER 'NONE'
SET TO VALUE</I></BLINK></H3>\n");
```

```
                                printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                                break;
                        }
                        else {
                                search[++searchidx].name = pairbuf[i].name;
                                search[searchidx].value = pairbuf[i].value;
/*                              printf("Name = %s, Value = %s\n", search[searchidx].name,
search[searchidx].value);*/
                        }
                }
                /* Parameter FROM value empty? */
                else if(((i % 3) == 1) && (none_flag == 0) && !strcmp(pairbuf[i].value,
"\0")) {
                                printf("<H3><BLINK><I>ERROR - PARAMETER 'FROM' VALUE
EMPTY</I></BLINK></H3>\n");
                                printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                                break;
                }
        }
        /*                              */
        /*                              */
        /* Develop the search criteria  */
        /*                              */
        /*                              */
        fprintf(fd,"Boolean = %d\n", boolean);
        if(searchidx >= 1) {
                fprintf(fd,"There are %d search conditions to satisfy\n\n", ((searchidx +1)/2));
                for(i=0;i<=searchidx;i++) {  /* Output the non NONE values used for DB
search */
                        fprintf(fd,"%s = %s\n", search[i].name, search[i].value);
                }
                fprintf(fd,"\n**************************************\n");
                est_criteria(search, searchidx, criteria);
                i=0;
                while(criteria[i].name != NULL) {
                        fprintf(fd,"%s = %s, %s\n", criteria[i].name, criteria[i].from,
criteria[i].to);
                        /* Perform error checking */
                        if(criteria[i].to == NULL) criteria[i].to = "0.0";
                        if(atof(criteria[i].to) < atof(criteria[i].from)) {
                                /* TO value equal zero? */
                                if(!(criteria[i].to = "0.0")) {
                                        printf("<H3><BLINK><I>ERROR - TO VALUE LESS
THAN FROM VALUE</I></BLINK></H3>\n");
```

```c
                              printf("<H3><BLINK><I>PLEASE TRY
AGAIN</I></BLINK></H3>\n");
                      }
              }
/*                    printf("Name = %s,From = %s,To = %s\n",criteria[i].name,
criteria[i].from, criteria[i].to);*/
                      i++;
      }

              /*                    */
              /*                    */
              /* Search the header files     */
              /*                    */
              /*                    */
              sprintf(list_name,"%ilist",getpid());  /* Create the list output CPI file name */
              sprintf(cmd,"/tmp/%s",list_name);
              hdr_search(criteria,hdr_path,cpi_path,fd,cmd,boolean);
              /*                    */
              /*                    */
              /* Output the search results    */
              /*                    */
              /*                    */
              system("cat /tmp/query_search >> /tmp/query_search_cum");
              system("cat /tmp/query_search | /usr/lib/sendmail -t
cavov@magnum.oc.rf.af.mil");
              fclose(fd); /* Close the query search file */
              /*
              Bug: using ! after > causes file to  be created but not written to. Just use >
              */
              /*       sprintf(cmd,"ls /xbox/disk0/cavovn/data | grep mat >
/tmp/%s",list_name);*/
              /*sprintf(cmd,"ls /xbox/disk0/cavovn/httpd/data | grep mat >
/tmp/%s",list_name);
              system(cmd);*/
/*            printf("<H3><BLINK><I>MATCHING MATLAB
FILES</I></BLINK></H3>\n"); */
              printf("<H3><I>MATCHING MATLAB FILES</I></H3>\n");
              printf("<P><STRONG>Please note that files ending with suffix 'gz' were \
              compressed using the utility 'gzip'. You will need 'gunzip' to uncompress \
              them.</STRONG><BR><BR>\n");
/*            printf("<FORM METHOD=\"POST\">\n");
              printf("<SELECT NAME=\"MCARM_Query_Results\" SIZE=10>\n");
              printf("<OPTION SELECTED>None\n"); */
              sprintf(cmd,"/tmp/%s",list_name);
              if((fd_cpi = fopen(cmd,"r")) == '\0') {
                      if(no_rec_found) { /* Matching records or files found? */
```

72

```c
                                printf("main: Cannot open '%s' file\n", list_name);
                                exit();
                        }
                }
                else {
                        counter = 0;
                        printf("<PRE>");
                        while((fgets(line,PATHLEN,fd_cpi)) != NULL) {
                                for(i=0;line[i] != NULL;i++)  /* Get rid of newline character */
                                        if(line[i] == '\n')
                                                line[i] = '\0'; /* Replace newline with NULL
character */
                                counter = counter + 1;
                                strcpy(buff,(strrchr(line,'/')+1)); /* Strip off path information */
                                if(counter%MODULO_OP == 0)
                                {
                                        printf("<A
HREF=\"/~cavovn/cpi/%s\">%s</A>\n",buff, buff);
                                }
                                else
                                {
                                        printf("<A HREF=\"/~cavovn/cpi/%s\">%s</A>
",buff, buff);
                                }
/*                              printf("<OPTION><A
HREF=\"/~cavovn/cpi/%s\">%s</A>",buff, buff); */
/*                              printf("<OPTION><A
HREF=\"/~cavovn/cpi/%s\">%s</A><BR>",buff, buff); */
                        }
                        printf("</PRE>");
/*                      printf("</SELECT>\n</FORM>\n"); */
                }
                if(!no_rec_found) {  /* No matching records or files found? */
                        printf("<B><BLINK><I>NO MATCH</I></BLINK></B>\n");
                }
                else {
                        printf("<B><I>%d Data Files Found</I></B>\n",no_rec_found);
                }
/*              printf("<H3><BLINK><I>Thank you. Please select compressed Matlab file
to download.</I></BLINK></H3>\n"); */
                printf("<H3><I>Thank you. Please select compressed Matlab file to
download.</I></H3>\n");
/*              printf("<H3><BLINK><I>Select <EM>Back</EM> pointer to return to
query.</I></BLINK></H3>\n"); */
                printf("<H3><I>Select <EM>Back</EM> pointer to return to
query.</I></H3>\n");
```

```
                fclose(fd_cpi);
        }
        else {
                printf("<H3><BLINK><I>NOTHING TO
SEARCH</I></BLINK></H3>\n");
        }
        printf("<a
href=\"http://sunrise.oc.rl.af.mil:80/~cavovn/forms/query_new.html\"><img align=\"left\"
src=\"/icons/point_11.gif\" alt=\"Back\"></a>\n");
        printf("</BODY></HTML>\n");
        system("/xbox/disk0/cavovn/bin/netscape
http://sunrise.oc.rl.af.mil/flight_path5.html");
/*
        Cleanup
*/
/*      sprintf(cmd,"rm -f /tmp/%s",list_name);*/
        system(cmd);    /* Remove the created list file */
}


split(buf,begin,end)
char buf[STRLEN]; /* stdin buffer */
int begin; /* beginning index pointer for name/value pair */
int end; /* end index pointer for name/value pair */
{
        register int i;

        for(i=begin;i<end;i++) {
                if(buf[i] == '=') {
                        buf[i] = '\0'; /* set equal to null to terminate name */
                        pairbuf[pairidx].name = &buf[begin];
                        pairbuf[pairidx].value = &buf[i+1];
                        return;
                }
        }
}

pack(buf,length)
char buf[STRLEN];
int length; /* string length in bytes */
{
        char temp[STRLEN];
        int i,j=0;

        for(i=0;i<length && buf[i] != '\0';i++) {
                if(buf[i] == '?') continue; /* Ignore hexdecimal symbol */
                else if(buf[i] != ' ')
```

74

```
                        temp[j++] = buf[i];
                else {
                        if(buf[i+1] == ' ')
                                continue;
                        else
                                temp[j++] = buf[i];
                }
        }
        temp[j] = '\0'; /* Null terminate temporary buffer */
        strcpy(buf,temp); /* Copy packed string into processing buffer */
}


est_criteria(inbuf, count, outbuf)
struct pairs inbuf[MAX_PAIRS];
int count;
struct triples outbuf[MAX_PAIRS];
{
        int i, j;

        j = 0;
        for(i=0;i<count;i=i+2) {
                outbuf[j].name = inbuf[i].value; /* Get the parameter name */
                outbuf[j].from = inbuf[i+1].value; /* Get the from value */
                outbuf[j++].to = inbuf[i+2].value; /* Get the to value */
        }
}


hdr_search(criteria,hdr_path,cpi_path,fd,cpi_file,boolean)
struct triples criteria[];
char hdr_path[],cpi_path[];
FILE *fd;
char cpi_file[MAX_PAIRS];
unsigned short boolean;
{
FILE *fd_hdr;
char hdr_list[PATHLEN], cmd[MAX_PAIRS], line[PATHLEN], buff[PATHLEN],
file_name[TEXT_SIZE];
int i, len;

        sprintf(hdr_list,"%ihdr_list",getpid());  /* Construct the header directory list output
file name */
        /* Imbedded newline in HDR_PATH caused must grief and  havoc. Causes system
'cmd' not to work! */
        sprintf(cmd,"ls %s | grep \"\\.hdr\" > /tmp/%s",hdr_path, hdr_list);
        system(cmd);
        sprintf(cmd,"/tmp/%s",hdr_list);
```

```
            fprintf(fd,"HDR_SEARCH %s\n",hdr_path);
            fprintf(fd,"HDR_SEARCH %s\n",cmd);
            if((fd_hdr = fopen(cmd,"r")) == '\0') {
                    printf("hdr_search: Cannot open '%s' file\n", hdr_list);
                    exit();
            }
            while((fgets(line,PATHLEN,fd_hdr)) != NULL) {
                    fprintf(fd,"HDR_SEARCH %s\n",line);
                    for(i=0;line[i] != NULL;i++)  /* Get rid of newline character */
                            if(line[i] == '\n')
                                    line[i] = '\0'; /* Replace newline with NULL character */
                    sprintf(file_name,"%s/%s",hdr_path,line); /* Construct header absolute file
name */
                    /* read_hdr routine returns 0 if no match, 1 if match */
                    if(read_hdr(criteria,file_name,boolean)) { /* Search header file for criteria
match */
                            strncpy(buff,line,(strcspn(line,"mat")-1));
/*                          sprintf(cmd,"ls %s/%s*CPI* >> %s",cpi_path, buff, cpi_file); */
                            sprintf(cmd,"ls %s/%s*.gz >> %s",cpi_path, buff, cpi_file);
                            ++no_rec_found; /* Increment number records found counter */
                            system(cmd);
                    }
                    len = strlen(buff);
                    for(i=0;i<len;i++) buff[i] = '\0'; /* Clear the buffer */
            }
            fclose(fd_hdr); /* Close the header list file */
            sprintf(cmd,"rm /tmp/%s",hdr_list);
            system(cmd);    /* Remove the created header list file */
}
```

*MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)*

The advancement and application of information systems science and

technology for aerospace command and control and its transition to air,

space, and ground systems to meet customer needs in the areas of Global

Awareness, Dynamic Planning and Execution, and Global Information

Exchange is the focus of this AFRL organization. The directorate's areas

of investigation include a broad spectrum of information and fusion,

communication, collaborative environment and modeling and simulation,

defensive information warfare, and intelligent information systems

technologies.